INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

On the Quadratic Assignment Problem and its Extensions

BY

Chawki A. FEDJKI

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

Systems Engineering

December 2000

UMI Number: 3000282

UMI Microform 3000282

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

Deanship of Graduate Studies

This Dissertation, written by **Chawki Abdulabaki FEDJKI** under the directions of his dissertation advisor **Professor Salah O. Duffuaa** and approved by his dissertation committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requiremants for the degree of

DOCTOR of PHILOSOPHY

in

SYSTEMS ENGINEERING

Dissertation Committee

20/2/2001
Prof. Salih O. Duffuaa,
Chairman

20/2/2001
Prof. Shokri Z. Selim,
Member

20/2/2001
Prof. Mohammed BenDaya,
Member

20/2/2001
Dr. Abdulbasit A. Andijani
Member

20/2/2001
Dr. Mohamad Al-Suwaiyel
Member

20/2/2001
Dr. Omar M. Al-Turki
Chairman, Department of
Systems Engineering

Prof. Osama A. Jannadi
Dean of Graduate Studies

21/2/2001
Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

Dedications

To my father and my Mother

To my Wife

To my Children Kawthar
And
Abdulrahim

I dedicate this humble work.

Acknowledgments

All praise be to ALLAH for his limitless help and guidance. Peace and blessings of ALLAH be upon his messenger Mohammad.

Acknowledgment is due to King Fahd University of Petroleum and Minerals for the generous help and support for this research.

I would like to express to my dissertation advisor, Professor Salih Osman Duffuaa, all my profound gratitude and my sincere appreciation for his guidance and patience throughout this research. His continuous support and encouragement will never be forgotten. Special thanks go to the members of the committee, Pr. Shokri Z. Selim, Pr. Mohammed BenDaya, Dr. Abdulbasit A. Andijani and Dr. Mohammed Al-Suwaiyel, for their consistent support and valuable suggestions. I would like to thank Professor Hanif D. Sherali for his valuable comments and suggestions. Also I would like to thank the faculty and staff members of Systems Engineering Department for their kindness and support. Finally, special thanks go to my family for their patience, encouragements, understanding and support during hard times I went through.

TABLE OF CONTENTS

خلاصة الأطروحة

الاســـــــم : شوقي بن عبد الباقي فجكي

عنوان الأطروحة : مسألة التخصيص التربيعية و تعميماتها

التخصص الدقيق : هندسة نظم

التاريـــــــخ : ديسمبر ٢٠٠٠

إن مسألة التخصيص التربيعية أكثر مسائل إيجاد الحلول المثلى التوافقية رواجاً، ولها العديد من التطبيقات في علوم الحاسب الآلي وبحوث العمليات. في هذه الأطروحة، تطرق لعدة جوانب من مسألة التخصيص التربيعية وامتداداتها. فيما يخص مسألة التخصيص التربيعية، نبدأ أولاً باستقصاء خصائص أدنى النقاط النجمية وانطلاقاً من هذه الخصائص والهيكل الشبكي الأساسي للمسألة نقترح خوارزمية أساسها النقاط الطرفية للمسألة. ثم نعيد صياغة المسألة كنموذج تخصيص خطي مقيدة و نبحث إمكانية استعمال هذه الصيغة في تطوير خوارزميات لمسألة التخصيص التربيعية. بعدها نقوم بتعميم مسألة التخصيص التربيعية حيث أن الصيغة الحالية لا تغطي جميع مسائل تحديد المواقع. ثم ندرس خصائص المسألة المقترحة ونبرهن أن مسألة التخصيص التربيعية وتعميماتها المعروفة هي حالة خاصة من مسألة التي اقترحناها.

تركز الأطروحة على حل حالة خاصة من المسألة الجديدة. نرمز لهذه الحالة الخاصة بمسألة شبه التخصيص التربيعية العامة. ثم نطور عدة خوارزميات تفريع وتحديد بتكييف حد جيلمور المعروف وباستعمال طريقة الاسترخاء الخطي للشرعلي وأدمس وذلك لحل مسألة شبه التخصيص التربيعية العامة. نولد مسائل اختبار معيارية بتعديل بعض مسائل الاختبار المعيارية المستخدمة في مسألة التخصيص التربيعية. توضح النتائج أن الحد المحرز بطريقة الاسترخاء الخطي أفضل بكثير من باقي الحدود ويستلزم أقل عدد من الفروع. غير أنه يحتاج إلى وقت طويل لحسابه مقارنتا بحد جيلمور، لذلك فإن أداء الخوارزميات المؤسسة على حد جيلمور كان أفضل من حيث وقت التنفيذ. وقد تم حل مسائل بحجم ٢٢ مرفق. وختمت الأطروحة، باقتراح اتجاهات لمواصلة البحث في هذه المسألة.

دكتوراه

جامعة الملك فهد للبترول والمعادن

الظهران، المملكة العربية السعودية

ديسمبر ٢٠٠٠

Abstract

Title : On the Quadratic Assignment Problem and its Extensions

Name : Chawki A. FEDJKI

Major : Systems Engineering

Date : December 2000

The quadratic assignment problem (QAP) is a well known combinatorial optimization problem with many applications in computer science and operations research. In this dissertation, we address several aspects of the QAP and some of its extensions. Concerning the QAP, the characterization of a local star minimum is first investigated. Based on this characterization and the underlying network structure of the QAP an extreme point based algorithm is proposed. Then the QAP is reformulated as a constrained linear assignment problem and the possibilities of using this formulation for developing algorithms for the QAP is investigated.
A generalization of the QAP formulation is given since the current formulation does not capture the general form of the facility location problem. The characteristics of the new problem are studied and the new formulation is shown to capture the QAP and its generalizations as special cases. In the rest of the dissertation, the focus was on solving a special case of the new problem. This case is termed the Generalized Quadratic Semi-Assignment Problem (GQSAP). Several branch and bound algorithms were developed based on the adaptation of the well known Gilmore's bound, and the Relaxation Linearization Technique (RLT) of Sherali and Adams to solve the GQSAP. A set of benchmark problems for this class is obtained by modifying existing Quadratic Assignment benchmark problems.
The results show that the RLT lower bound outperforms the other bounds thus requires a smaller branching tree. On the other hand and due to its high computation time in comparison with the Gilmore's bound, algorithms based on the later performed better in terms of running time. Problems of size up to 22 facilities were solved. The dissertation is concluded with directions for further research.

DOCTOR of PHILOSOPHY DEGREE

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia

December 2000

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Statement of the problem

The quadratic assignment problem ($\mathcal{QAP}$) is a well known combinatorial optimization problem. It gained its interest for the many practical applications in operations research and computer science. Examples of areas of application of the $\mathcal{QAP}$ are: Facility layout and design, capital budgeting, resource allocation, communication network design, $VLSI$ design and scheduling, and many more.
Section 2.6 provides more detailed examples of areas of application of the $\mathcal{QAP}$ found in the literature.

The $\mathcal{QAP}$ was first introduced in 1958 by Koopmans and Beckmann [80]. It is stated as:

Given n facilities and n locations, with known flows among facilities, known distances

among locations and with a known cost of facility location. The objective is to find a one to one assignment of facilities to locations that would minimize the total flow-distance interaction and assignment cost.

Formally, the problem can be stated as:

Find a permutation π of the element of the set $S = \{1, 2, 3, ..., n\}$ that minimizes the cost of the permutation. The cost is expressed as follows:

P 1.1

$$\min_{\pi \in P_n} Cost(\pi) = \min_{\pi \in P_n} \left[\sum_{i=1}^{n} c_{i\pi(i)} + \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)} \right]$$

Where:

P_n : is the set of all possible permutations of n elements

f_{ij} : is the flow from facility i to facility j.

d_{kl} : is the distance from locations k to location l.

c_{ij} : is the cost of assigning facility i to location j.

An equivalent formulation of the above problem can be given, where the problem is stated rather analytically than in its combinatorial form.

Let

$$x_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is located at location } j \\ 0 & \text{otherwise} \end{cases}$$

The formulation is given as follows:

P 1.2

$$\textit{minimize} \qquad Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}x_{ij}x_{kl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n$$

$$x_{ij} \in \{0, 1\}$$

The $\mathcal{QAP}$ turned out to be computationally very difficult. This difficulty is mainly due to the quadratic form of the objective function in addition to its combinatorial nature. The difficulty of the $\mathcal{QAP}$ is illustrated in the following theorems due to Sahni and Gonzalez [115]. Before stating the theorem, let us start with some definitions.

Definition 1 *A problem π is said to be in the class $\mathcal{NP}$, if for every feasible solution there is a polynomial way to verify that the solution is feasible.*

Definition 2 *A problem π is said to be $\mathcal{NP}$-hard if all problems in the class $\mathcal{NP}$ polynomially reduce to π.*

Definition 3 *Given a real number $\epsilon > 0$, an algorithm $\mathcal{A}$ for the $\mathcal{QAP}$ is said to be an ϵ-approximation algorithm if and only if*

$$\left|\frac{Cost(\pi_A) - Cost(\pi_{opt})}{Cost(\pi_{opt})}\right| \leq \epsilon$$

Where π_A is a permutation obtained by algorithm $\mathcal{A}$ and π_{opt} is an optimal per-

mutation. The solution produced by an ϵ-approximation algorithm is called an ϵ-approximate solution.

Theorem 1 *The $\mathcal{QAP}$ is $\mathcal{NP}$-hard.*

Proof: (See Sahni and Gonzalez [115])

In fact not only solving the $\mathcal{QAP}$ was found to be very difficult, but even finding an ϵ-approximate solution of it, is very difficult. This result due also to Sahni and Gonzalez is stated in the following theorem.

Theorem 2 *For an arbitrary $\epsilon > 0$, the problem of finding an ϵ-approximate solution to the $\mathcal{QAP}$ is $\mathcal{NP}$-hard.*

Proof. (See Sahni and Gonzalez [115])

1.2 Objectives of the research

Many approaches were used to solve the $\mathcal{QAP}$ problem. These approaches range from implicit enumeration to recent Meta-heuristics including decomposition and construction-and-improvement methods. With all of the huge effort that has been spent on the problem throughout the years, only problems of sizes not exceeding 22 facilities are reported to have been solved [24]. It is reported in a private communication [24] that the Nugent problem [98] of size 25 have been solved optimally. Researchers have tried to explore other ways of dealing with such a hard problem by

trying to obtain more tractable reformulations and use known efficient methods to solve the resulting formulations. These reformulations try to deal with the hardness induced by the nonlinearity of the objective function, leading in general to a linearizations of the problem at the cost of lifting the dimension of the problem. These reformulations failed, in general, when dealing with large size problems. On the other hand, many extensions of the original $\mathcal{QAP}$ were studied. The literature indicates that this is still an active area of research, this indication motivated our interest in the subject. Based on the above motivations, four objectives have been formulated to address in this dissertation. The objectives are given in sections 1.2.1 to 1.2.4.

1.2.1 Investigate the network structure of the $\mathcal{QAP}$.

One feature of the $\mathcal{QAP}$, is that its constraint matrix is unimodular. This fact motivated us to deal with the problem as a network problem. Using the idea of the network simplex method, we give a characterization of a local minimum, with respect to a given basis, for the $\mathcal{QAP}$, an then discuss ways of using the results obtained in designing what should represent efficient heuristics to solve the problem.

1.2.2 A new formulation for the problem

The second objective of the research is to try to look at the problem from a different angle. This led to a linear formulation of the $\mathcal{QAP}$. This formulation is obtained from a graphical transformation of the problem. This transformation may

later be studied for designing new algorithms for the $\mathcal{QAP}$ that could be more efficient than those available at this time.

1.2.3 Generalization of the $\mathcal{QAP}$.

As a third objective in this dissertation, it was observed that in all the extensions of the $\mathcal{QAP}$, the most general form of facility location problems is not addressed. The actual formulations have many limitations that restrict their applications. These limitations include:

- Only one facility can be assigned to a location. This limits the opportunity of locating several facilities at a single location. Although this limitation may be overcome with the actual formulations, by duplicating such locations as many times as needed, this would increase the size of the problem and hence limit the possibility of solving large problems. The largest size of $\mathcal{QAP}$ that can be solved in a reasonable amount of time does not exceed a size of 22, as reported in the literature.

- The number of facilities to be assigned to each location is fixed. This number is not determined through the optimization process in case where some locations may accommodate more facilities than specified in $\mathcal{QAP}$.

Allowing this to happen may complicate the problem, but it is expected to enrich the applications and identify new opportunities for optimization. Taking into consid-

eration the above mentioned limitations of the standard $\mathcal{QAP}$ led to a new problem that is proposed and studied in this research.

1.2.4 Algorithms for a special case of the generalization.

Finally, and as a first trial to solve the proposed generalization of the $\mathcal{QAP}$, we develop and test many branch and bound algorithms to solve a special case of the new generalization. The first set of algorithms are based on adaptations of the well known Gilmore's bounds. The second set of branch and bound algorithms are based on a bound obtained using the Relaxation Linearization Technique ($\mathcal{RLT}$) of Sherali and Adams[123]

1.3 Dissertation Organization

In the next chapter an overview of the literature of the $\mathcal{QAP}$ is given. In Chapter 3, some conditions to characterize local minima for the $\mathcal{QAP}$ are provided. The chapter is concluded by discussing further research work that may be done using the results obtained. In Chapter 4 a transformation of the $\mathcal{QAP}$ is derived and a mixed integer linear formulation is proposed. Possible ways of using this transformation to solve the $\mathcal{QAP}$ are also discussed in this chapter. Chapter 5 is concerned with the generalization of the $\mathcal{QAP}$, where a new problem together with some of its special cases are introduced, these represent a more general class of problems in the area of facility location. In Chapter 6, several branch and bound algorithm for a special

case of the general problem are presented. The algorithms proposed are based on an adaptation of different versions of the well known Gilmore's bounds for the $\mathcal{QAP}$. In this chapter, six branch and bound algorithms are developed and tested. The chapter is concluded by presenting and discussing the results obtained. In Chapter 7, other branch and bound algorithms, for the same special case problem are proposed and the results are discussed, the bound in this case is obtained using the $\mathcal{RLT}$ of Sherali and Adams [123]. The performance of these algorithms compared to the best one in the previous chapter is investigated. Finally, Chapter 8 concludes the dissertation by outlining further areas of research.

Chapter 2

Literature review

2.1 Introduction

In this chapter an attempt is made to review the vast literature on the $\mathcal{QAP}$. The chapter is started by examining the characteristics of the assignment polytope, followed by the different formulations and extensions of the problem. Because transformations of the problem such as linearization may be considered rather as tentative solutions than formulations, these cases are discussed in a separate section that is called: problem transformation. Next, some well known problems that are somewhat related to the $\mathcal{QAP}$ are presented. Then the applications of the quadratic assignment problem are discussed. In reality, it is this wide range of applications in addition to the challenge presented to researchers that are behind the interest given to the $\mathcal{QAP}$ during the last four decades. Exact and heuristic approaches developed over the

years, for this challenging problem, are outlined next. The chapter is ended with a brief conclusion.

2.2 Characteristics of the assignment polytope

A comprehensive study of the assignment polytope due to Balinski and Russakoff [6] was summarized by Sherali [120], and provided a set of properties for the assignment polytope that is defined by :

$$\begin{aligned} &\textstyle\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n \\ &\textstyle\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n \\ &x_{ij} \geq 0 \end{aligned} \tag{2.1}$$

In other word, we refer to the convex hull of the assignment feasible set as the assignment polytope. Before listing these properties, let us give the following definitions:

Definition 4 *Given a graph G. A path between two nodes x and y of a graph G is a set of edges (i_{k-1}, i_k), $k = 1, ..., t$ such that $x = i_0$ and $y = i_t$. A circuit in a graph is a path that starts and ends at the same node. A Hamiltonian circuit in a graph G is a circuit that passes through all the nodes of the graph. The distance between two nodes of a graph G is the minimum number of edges in any path linking them. The diameter of a graph G is the maximum distance between any pair of nodes of the graph. The girth of a graph G is the minimum number of edges in any circuit of*

the graph. The circumference of a graph G is the maximum number of edges in any circuit of the graph.

Definition 5 *Two feasible bases of the assignment polytope are said to be neighbors if the cardinality of the intersection of their respective basis sets (spanning trees) is $(2n-2)$, where n is the number of facilities/locations to be assigned.*

Definition 6 *Two extreme points $x \neq y$ of the assignment polytope are said to be neighbors if they have feasible basis representing them that are neighbors.*

Definition 7 *Let $G(P)$ be the graph whose nodes are the extreme points of the polytope P, and for any two nodes x and y of $G(P)$, the edge $(x, y) \in G(P)$ if and only if x and y are neighbors.*

Definition 8 *The valency of a node of $G(P)$ is the number of edges incident to it. In other words the valency of a node x is the number of neighboring nodes of x.*

Definition 9 *Given an extreme point x of the assignment polytope, let $B(x)$ be the subgraph of the complete assignment which has arcs corresponding only to nonzero variables*

Property 1: Two bases of the assignment polytope are neighbors if and only if the union of the spanning trees representing them contains exactly one cycle.

Property 2: Two extreme points $x \neq y$ of the assignment polytope are neighbors if and only if $B(x) \cup B(y)$ contains exactly one cycle.

Property 3: The graph $G(P)$ has the following properties:

(a) $G(P)$ has $n!$ nodes.

(b) Any node x of $G(P)$ has valency $N = \sum_{k=0}^{n-2} \binom{n}{k} (n-k-1)!$

(c) $G(P)$ has diameter 2. That is any two nodes of $G(P)$ are either neighbors or there is a third node to which they are both neighbors.

(d) $G(P)$ contains a Hamiltonian circuit.

(e) $G(P)$ has girth of 3 and circumference of $n!$

Property 4: To each extreme point of the assignment polytope there correspond $(2^{n-1})(n^{n-2})$ feasible bases. Further, The polytope has $(2^{n-1})(n^{n-2})n!$ feasible bases and $n^{2(n-1)}$ bases.

Property 5: Given any two pair of feasible bases, a path of neighboring feasible bases of length at most $(2n-1)$ connects them. This statement was strengthened by Sherali [120] as follows: Given any pair of extreme points $x \neq y$, if the intersection $B(x) \cap B(y)$ has p components of which q are circuits, then a path of neighboring feasible bases of length at most $[2n-2(p-q)-1]$ connects them.

Property 6: The assignment polytope has dimension $(n-1)^2$ and n^2 facets.

The following table, provides some statistics of the assignment polytope.

Size	Dimension	Number of			
n	$(n-1)^2$	Facets	Extrm. pts	Neibr. extrm. pts	Bases/extrm. pt
2	1	4	2	1	2
3	4	9	6	5	12
4	9	16	24	20	144
5	16	25	120	84	2000
6	25	36	720	409	41472
7	36	49	5040	2540	1075648
8	49	64	40320	16064	33554432

Table 2.1: Statistics on the assignment polytope

2.3 Alternative formulations and extensions.

In this section the different formulations of the problem are presented, followed by various extensions obtained for the original problem [80] introduced in 1957.

2.3.1 Alternative formulations

Although the combinatorial structure of the $\mathcal{QAP}$ is better expressed by the formulation given in P 1.1, other mathematical formulations have been studied and were found to be very efficient in designing algorithms for the $\mathcal{QAP}$. Indeed, it is one of these formulations that we will deal with throughout this research.

Koopmans and Beckmann formulation

The formulation that was initially given by Koopmans and Beckmann [80] relies on the one to one correspondence between the set of permutations P_n and the assignment feasible set. The assignment feasible set being given by the following

$$\begin{aligned} &\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1, 2, ..., n\} \\ &\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1, 2, ..., n\} \\ &x_{ij} \in \{0, 1\} \end{aligned} \tag{2.2}$$

This correspondence led to the following formulation of the $\mathcal{QAP}$.

P 2.1

$$\begin{aligned} &\textit{minimize} \qquad Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} f_{il}d_{jm}x_{ij}x_{lm} \\ &\textit{subject to} \ : \\ &\qquad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n \\ &\qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n \\ &\qquad x_{ij} \in \{0, 1\} \end{aligned}$$

Where:

$$x_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is located at location } j \\ 0 & \text{otherwise} \end{cases}$$

In this research, the interest is on the latter formulation. Here the linear term is dropped from consideration as it does not add any significance to the complexity of

the problem. Also, all the results obtained can easily be adopted to the case where the linear term is added.

Trace formulation

Another alternative formulation was derived by introducing the so-called permutation matrices and the notion of trace of a matrix that we define below.

Definition 10 *A permutation matrix is a square matrix P, with boolean entries (i.e. $p_{ij} \in \{0,1\}$), and such that each row and each column contains a single nonzero entry. In other words, it can be considered as the identity matrix with rearranged rows (columns).*

Definition 11 *The trace of a matrix $A = (a_{ij})_{i,j=1,n}$ is the sum of the diagonal elements of A. That is $Tr(A) = \sum_{i=1,n} a_{ii}$.*

Given the above definitions, the trace formulation of the quadratic assignment problem is given as:

P 2.2

$$\underset{P \in \pi_n}{minimize} \; Tr(FPDP^t)$$

Where F is an $n \times n$ matrix whose entries f_{ij} consist of the flow between facilities i and j, and D is an $n \times n$ matrix whose entries d_{kl} consist of the distance between locations k and l. The set π_n represents the set of all possible permutation matrices of order n.

Kronecker product formulation

Another alternative matrix representation of the $\mathcal{QAP}$ is given using the Kronecker product, that is also known as the *tensor product.* The Kronecker product is defined bellow.

Definition 12 *Given two matrices A $(m \times n)$ and B $(p \times q)$, the tensor product or Kronecker product of A and B is a matrix formed by all possible products of elements of A and B. In other words, if we use the symbol $\otimes$ to represent the Kronecker product then we have:*

$$A \otimes B = (a_{ij}B) = (a_{ij}b_{kl}) \qquad i = 1, m; j = 1, n; k = 1, p; l = 1, q$$

It is clear that the Hessian of the objective function of the $\mathcal{QAP}$ represents the Kronecker product of the flow and distance matrices. Then, if we consider that X_P is the n^2 vector whose elements are the columns of a permutation matrix $P \in \pi_n$, the objective function of the $\mathcal{QAP}$ can be written as:

P 2.3

$$\underset{X_P, P \in \pi_n}{minimize} \; Z = X_P^t (F \otimes D) X_P$$

The two last formulations above use the notion of eigenvalues to deal with the $\mathcal{QAP}$.

2.3.2 Extensions of the $\mathcal{QAP}$

In 1963, Lawler [86] introduced two extensions to the $\mathcal{QAP}$. In the first extension, he introduced a general term to represent the objects interactions. In this case, instead of the two $n \times n$ matrices representing the flows and distances respectively, a four dimensional square matrix of order n with elements d_{ijkl} is used. This provides a more general range of application to the $\mathcal{QAP}$.

This led to the following problem:

P 2.4

$$\textit{minimize} \qquad Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} d_{ijlm}x_{ij}x_{lm}$$

$$\textit{subject to} \quad :$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n$$

$$x_{ij} \in \{0, 1\}$$

In the second extension, the multi-commodity case was taken into consideration. In this case, the flow not only depends on the facilities but also on the type of commodity to be transported. Hence the usual flow is replaced by f_{ik}^{n} for $n = 1, ..., p$, where f_{ik}^{n} corresponds to the flow of commodity n from facility i to facility k. Similarly, d_{jl}^{n} corresponds to the cost per unit flow of product n from location j to location l.

This new consideration led to the following problem:

P 2.5

$$\textit{minimize} \qquad Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} f_{il}^{k}d_{jm}^{k}x_{ij}x_{lm}$$

$$\textit{subject to} \ :$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n$$

$$x_{ij} \in \{0,1\}$$

In 1970, Graves and Whinston [58] introduced another term w_{ijkl} to the objective function. The added term would depend on a pair of assignment. Assuming, that such term exists for each commodity k, $k = 1, ..., p$. We obtain a more general form of the $\mathcal{QAP}$ where the objective function has the following form:

$$\begin{aligned} \textit{minimize } Z(x) \ = \ & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} f_{il}^{k}d_{jm}^{k}x_{ij}x_{lm} \\ & + \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} w_{ijkm}^{k}x_{ij}x_{lm} \end{aligned}$$

A more compact transformation introduced by Lawler [86] and extended by Pierce and Crowston [106] provide a more convenient way of representing the $\mathcal{QAP}$. The transformation introduces a term s_{ijkl} that represents the overall pairwise interaction. So, if we let s_{ijkl} be such that

$$s_{ijkl} = \begin{cases} \sum_{t=1}^{p} w_{ijkl}^{t} + \sum_{t=1}^{p} f_{ik}^{t}d_{jl}^{t} & i \neq k \text{ or } j \neq l \\ c_{ij} + \sum_{t=1}^{p} w_{ijij}^{t} & i \neq k \text{ or } j \neq l \end{cases}$$

We obtain the following problem:

P 2.6

$$\begin{aligned} \textit{minimize} \quad & Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} s_{ijkl}x_{ij}x_{kl} \\ \textit{subject to} \ : \quad & \\ & \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n \\ & \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n \\ & x_{ij} \in \{0,1\} \end{aligned}$$

2.4 Problem transformation

The quadratic form of the objective function of the $\mathcal{QAP}$ destroys any hope of finding good solution methods. This fact directed many researchers toward linearizing the problem. Kettani and Oral [79] propose a reformulation of the problem by linearizing the quadratic constraint then reducing the number of 0-1 variables. The linearization is done through the introduction of a new variable and then the quadratic objective function is linearized using bounds on this new variable. Yamada [142] proposes a new formulation using polynomial equations for a special class of $\mathcal{QAP}$, namely $\mathcal{QAP}$ on r-dimensional gird. Other linearizations have attracted more people, and are summarized below.

2.4.1 Kaufman and Broeckx

Considering the Koopmans and Beckmann's formulation (**P 2.1**), and rewriting the objective function as:

$$Z = \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ik} \sum_{j=1}^{n}\sum_{l=1}^{n} f_{ij}d_{kl}x_{jl}$$

Kaufman and Broeckx [77] introduced new n^2 variables w_{ik} such that:

$$w_{ik} = x_{ik} \sum_{j=1}^{n}\sum_{l=1}^{n} f_{ij}d_{kl}x_{jl} \qquad i,k = 1...n$$

Using these new variables, they obtained a mixed integer linear problem with $2n^2$ variables and $n^2 + 2n$ constraints. The new problem is presented bellow.

P 2.7

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{k=1}^{n} w_{ik}$$

$$\textit{subject to} \ :$$

$$c_{ik}x_{ik} + \sum_{j=1}^{n}\sum_{l=1}^{n} f_{ij}d_{kl}x_{jl} - w_{ik} \leq c_{ik} \qquad i,k = 1...n$$

$$\sum_{i=1}^{n} x_{ik} = 1 \qquad k = 1...n$$

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad i = 1...n$$

$$x_{ik} \in \{0,1\}, \qquad w_{ik} \geq 0 \qquad i,k = 1...n$$

where $c_{ik} = \sum_{j=1}^{n}\sum_{l=1}^{n} f_{ij}d_{kl}$, $i,k = 1...n$.

Kaufman and Broeckx [77] derived an algorithm using Bender's decomposition applied to the above transformation.

2.4.2 Bazaraa and Sherali

Bazaraa and Sherali [12] proposed a linearization to the general $\mathcal{QAP}$ given in (**P 2.6**). An order of $\frac{n^2(n-1)^2}{2}$ new variables y_{ijkl} are introduced replacing the products $x_{ij}x_{kl}$ leading to a linear objective function. Again the problem obtained this way is a mixed integer linear problem with $\frac{n^2(n-1)^2}{2}+n^2$ variables and $2n(n-1)$ constraints, that is, more variables and more constraints than the one obtained in (**P 2.7**). The problem is given below.

P 2.8

$$\textit{minimize} \quad \sum_{i=1}^{n-1}\sum_{j=1}^{n}\sum_{k=i+1}^{n}\sum_{l=1,l\neq j}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{k=i+1}^{n}\sum_{l=1}^{n} y_{ijkl} - (n-1)x_{ij} = 0 \qquad i = 1...n-1, j = 1...n$$

$$\sum_{i=1}^{k-1}\sum_{j=1,j\neq l}^{n} y_{ijkl} - (k-1)x_{ij} = 0 \quad k = 2,...,n-1, l = 1...n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$x_{jl} \in \{0,1\}, 0 \leq y_{ijkl} \leq 1 \qquad i = 1...n-1, k = i+1...n, j, l = 1...n$$

This transformation was used by Bazaraa and Sherali [12, 13] in algorithms using Bender's decomposition and cutting planes methods.

2.4.3 Frieze and Yedegar

Again, considering the Koopmans and Beckmann's formulation (**P 2.1**) of the $\mathcal{QAP}$, Frieze and Yedegar [51] introduced n^4 variables y_{ijkl} such that $y_{ijkl} = x_{ik}x_{jl}$, $1 \leq i, j, k, l \leq n$, they obtained the following Mixed Integer Linear Problem:

P 2.9

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{i=1}^{n} y_{ijkl} = x_{jl} \qquad j, k, l = 1...n$$

$$\sum_{j=1}^{n} y_{ijkl} = x_{ik} \qquad i, k, l = 1...n$$

$$\sum_{k=1}^{n} y_{ijkl} = x_{jl} \qquad i, j, l = 1...n$$

$$\sum_{l=1}^{n} y_{ijkl} = x_{ik} \qquad i, j, k = 1...n$$

$$y_{iikk} = x_{ik} \qquad i, k = 1...n$$

$$y_{ijkl} \leq 1 \qquad i, j, k, l = 1...n$$

$$x_{jl} \in \{0, 1\}, y_{ijkl} \geq 0 \qquad i, j, k, l = 1...n$$

This problem has n^4 real variables, n^2 boolean variables and $n^4 + 4n^3 + n^2 + 2n$ constraints. It was used to derive a lower bound for the $\mathcal{QAP}$. The number of real variables can be reduced by a half by observing that $y_{ijkl} = x_{ik}x_{jl} = x_{jl}x_{ik} = y_{jilk}$.

2.4.4 Adams and Johnson

Using the above observation, Adams and Johnson [1] modified the formulation of Frieze and Yedegar (**P 2.9**) to obtain a linearization of the $\mathcal{QAP}$ with less number of constraints as follows:

P 2.10

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{k=1 \\ k\neq i}}^{n} y_{ijkl} = x_{ij} \qquad i,j,l = 1...n, i \neq k$$

$$\sum_{\substack{l=1 \\ l\neq j}}^{n} y_{ijkl} = x_{ij} \qquad i,j,k = 1...n, j \neq l$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l = 1...n, i < k, l \neq j$$

$$x_{ij} \in \{0,1\}, y_{ijkl} \geq 0 \qquad i,j,k,l = 1...n, k \neq i, j \neq l$$

Adams and Johnson [1] argue that many of the existing lower bounds are directly related to the above problem (**P 2.10**). It is to be noted that the number of variables y_{ijkl} and constraints, in the formulation (**P 2.10**) may further be reduced by substituting $y_{ijkl} = y_{klij}$ into the objective function and the constraints.

2.5 Applications of the $\mathcal{QAP}$

In this section various examples of applications of the $\mathcal{QAP}$, found the literature, are outlined.

The list is non-exhaustive as many real-life problems are directly formulated or could be transformed into Quadratic Assignment Problems.

2.5.1 Facility layout

The quadratic assignment problem was first introduced by Koopmans and Beckmann [80] in an economic context, where a set of economic facilities are to be located in a set of predefined locations in order to minimize a quadratic cost function. This cost function depends both on a known flow between facilities and a known distance between the predefined locations. In fact, facility location is one of the major areas of application of the $\mathcal{QAP}$. Sarker et al. [117] formulated the problem of assigning m machines to m unique equally spaced locations along a linear material handling track, where the objective is to minimize the cost of backtracking. Lacksonen [81] used the $\mathcal{QAP}$ to solve a dynamic layout problem. Shang [119] used the quadratic assignment problem to solve a multi-criteria facility layout problem. Other work in the area of facility location may be found in Bazaraa[9], Chan and Francis [33], Francis and White [52], Fu and Kaku [53], Graves and Whinston [58], Hanan and Kurtzberg [64], Hillier [68], Ireland [72], Kaku and Rachamadugu [73], Moore [94], Seehof and Evans [118] and Vollmann and Buffa [132].

2.5.2 The wiring problem

Another area of application of the $\mathcal{QAP}$ is the wiring problem. In this problem, a number of modules are to be placed on a board. The modules are pairwise connected by a number of wires. The objective is to find a placement of the modules on the board, such that the total length of the wires is minimized. In this case, it can be assumed that the matrix F represents the number of wires connecting two modules and that the matrix D represents the distances between places on the board. This problem can be formulated as a $\mathcal{QAP}$.

Steinberg [127], described a concrete application of the $\mathcal{QAP}$ for the wiring problem, namely, the backboard wiring problem.

Other applications in this context may be found in Breuer [14], Pomentate [107].

2.5.3 Architectural design

Here the interest is about the interaction between rooms, or department rather than shipments between plants. The distance is usually rectilinear over a three dimensional space to take care of the case of multistory buildings. The distances between departments may as well be replaced by the time taken by employees in walking between departments. The case of hospital layout was addressed by Elshafei [47]. A campus planning model is presented by Dickey and Hopkins [43].

2.5.4 Other applications

Many other areas of application of the $\mathcal{QAP}$ may be found in the literature. In the area of scheduling, Geoffrin and Graves [55] considered the case of scheduling parallel production lines. Also, problems in the area of statistical data analysis have been formulated as quadratic assignment problem, such cases may be found in Burkov et al. [22], Hubert [70] and Timofeev and Litvinov [130]. The turbine balancing problem was also addressed as a $\mathcal{QAP}$. In this case, turbine blades are placed in equidistant positions along the circumference of a wheel. The variability in the weight of the blades causes the center of gravity of the assembled to deviate from its geometric center. The objective is to find a placement of the blades with minimum variation. Such applications may be found in Fathi and Ginjupalli [48], Laporte and Mercure [83] and Mosevich [95]. The airline gate assignment problem, where the concern is to allocate gates to aircraft at an airport in order to minimize passengers walking distances, was first suggested for further research as $\mathcal{QAP}$ by Mangoubi and Mathaisel [90]. This formulation was further extended and studied by Sherali and Brown [121] and later by Haghani and Chen [59].

2.6 Algorithms for the $\mathcal{QAP}$

In this section the existing algorithms for solving the quadratic assignment problem are briefly reviewed.

2.6.1 Exact Solutions

Most of the exact algorithms for the $\mathcal{QAP}$ are based on what is known as implicit enumeration or Branch and bound. This technique is based on two concepts, a controlled enumeration of all feasible solutions, and the elimination from consideration of solutions that are known to be unacceptable from dominance and feasibility criteria. For the $\mathcal{QAP}$, the difference between different branch and bound solutions lies on the choice of the branching procedure and the bounding technique used. Because of the importance of the bounding technique, it will be discussed separately. The branching strategies can be clustered into three different classes, namely single assignment, pair assignment and pair exclusion methods.

For the single assignment strategy, at each node of the tree, a facility is assigned to a location based on a selection criteria. This assignment remains unchanged for all descendent of the node where the assignment is made. Backtracking happens when it is found that descending further in that direction will not improve the solution at hand. Many selection rules have been proposed. One selection rule that is often used when the bounding technique solves a linear assignment problem ($\mathcal{LAP}$), is based on the so-called alternative costs. The alternative cost for a pair (i, j) is given by:

$$p_{ij} = \min\{\overline{c}_{ik} : k \neq j,\ 1 \leq k \leq n\} + \min\{\overline{c}_{ik} : k \neq i,\ 1 \leq k \leq n\}$$

where the $\overline{c}_{ij}$ are the reduced costs of the last $\mathcal{LAP}$ solved. In that context, Burkard [20] proposed a maximization of the alternative cost for selecting the pair of indices

(i, j). Mautor and Roucairol [92] proposed a different selection rule also based on the notion of alternative costs. Any pair for which the lower bound at the current node added to its alternative cost exceeds the upper bound at that node is marked as forbidden, and the index i appearing in the largest number of forbidden pairs is chosen. The index j is selected such that the pair (i, j) is not forbidden and the sum of the p_{kj} for non forbidden pairs (k, j) is maximized. In a different spirit, Bazaraa and Kirka [11] try to minimize the sum of all lower bounds at the next level of the branch and bound tree, and simultaneously minimize the number of branches from the next branch and bound tree level. Algorithms based on the single assignment strategy have been proposed by Gilmore [56], then generalized by Lawler [86], Burkard and Drigs [23], Edwards [45], Kaku and Thompson [74], Graves and Winston [58], Pardalos and Crouse [100], and Mautor and Roucairol [92].

The pair assignment strategy attempts to reduce the computer running time, using the fact that there are $m(m-1)$ pairs of facilities and $m(m-1)$ pairs of locations. Than at each node a pair of facilities is assigned to a pair of locations. Algorithms in this class have been proposed by Gavett and Plyter [54] and Land [82] for the symmetric $\mathcal{QAP}$, and by Pierce and Crowston [106] for the non symmetric case. The Gavett and Plyter [54] algorithm was improved later by Smith [126] using a parametric programming method.

Pair exclusion based algorithms proceeds on the basis of a stage by stage exclusion of assignment from a solution of the problem. Algorithm based on such idea have

been reported by Pierce and Crowston [106] and Smith [126].

2.6.2 Lower bounds

Lower bounds occupy a special place when reviewing solution techniques for the $\mathcal{QAP}$. This importance is mainly due to the difficulty of deriving good algorithmic solution not based on the branch and bound technique. The first bound derived for the $\mathcal{QAP}$ was proposed by Gilmore [56], it will be discussed in details in chapter 6. It is based on the idea that minimizing permuted dot products is obtained by simply rearranging the elements of the vectors involved. Since then, many bounds were proposed. In general, bounds for the $\mathcal{QAP}$ can be classified into four groups: Gilmore's related bounds, eigenvalue based bounds, lower bounds based on transformation and linearization, and lower bounds based on semi-definite relaxation.

Gilmore's related bounds

Gilmore [56] uses the idea of the permuted dot product to derive coefficients of the assignment variables, these coefficients are then used to solve a linear assignment problem ($\mathcal{LAP}$). This is the simplest and cheapest lower bound for the $\mathcal{QAP}$, it has $\mathcal{O}(n^3)$ time complexity for the Koopmans and Beckmann problem given in (**P 1.2**). A part from the bound proposed by Gilmore [56], in the literature there are several other bounds that try to generalize or strengthen the original bound of Gilmore or some of its derivatives. From these we have the Lawler's bound [86], in fact most of

the literature cite it as Gilmore-Lawler bound. This bound was developed independently by Gilmore[56] and Lawler [86]. In this case (n^2+1) $\mathcal{LAP}$ are solved to obtain the bound. The first n^2 $\mathcal{LAP}$ are solved to get the coefficients of the variables x_{ij} that are then used in the $(n^2+1)^{st}$ $\mathcal{LAP}$ to get the lower bound. This bound turned out to be equivalent to the Gilmore's bound for the Koopmans and Beckmann $\mathcal{QAP}$ (**P 1.2**), where only a single $\mathcal{LAP}$ is solved. It has an $\mathcal{O}(n^3)$ for the Koopmans and Beckmann problem (**P 1.2**) and $\mathcal{O}(n^5)$ for the Lawler's problem given in (**P 2.4**). Other authors improved the Gilmore's bound by transforming the coefficient matrices F and D to matrices with smaller entries. These are known as reduction methods. They were first introduced by Conrad [38], then further studied by Burkard [18, 20], Edwards [45], Fink et al. [49], Frieze and Yedegar [51] and Roucairol [113]. A more general, yet similar, bounding procedure was proposed by Christofides and Gerrard [36]. It is based on a decomposition of graphs obtained from the flow and distance matrices of the $\mathcal{QAP}$. Optimal reduction schemes used by Li et al. [87] led to a so-called variance reduction lower bound. A bounding procedure combining Gilmore-Lawler bound ideas with reduction steps was proposed by Hahn and Grant [62] and Hahn et al.[63].

Eigenvalue related bounds

These type of bounds use the eigenvalues associated with coefficients matrix of the trace formulation of the $\mathcal{QAP}$ (**P 2.2**). Although such bounds produced good

quality solutions, they are unpractical for use in a branch and bound scheme due to their high computation time. Bounds using such a framework have been proposed by Finke et al. [49], Hadley et al. [60, 61] Rendl and Wolkowicz [110].

Transformation and linearization based bounds

Assad and Xu [4] obtained an equivalent $\mathcal{QAP}$ by applying a transformation of the coefficients of the matrix in the Lawler formulation (**P 2.4**). The transformation is obtained by the introduction of new parameters. For a chosen setting of these parameters, a lower bound on the equivalent $\mathcal{QAP}$ is obtained using any known bounding technique (usually Gilmore-Lawler type bounds because they are the cheapest). The process is repeated iteratively producing non-decreasing lower bounds until a stopping criterion is met. The last bound obtained is a valid lower bound for the original $\mathcal{QAP}$. Such bounding technique has also been studied by Carraresi and Malucelli [28, 29] with a different setting for the parameters.

The different linearizations proposed in section 2.2 above have been used in obtaining lower bounds for the $\mathcal{QAP}$. But the one that provided the best bounds for many of the problems in the literature is the linearization of Adams and Johnson (**P 2.10**). Such linearization have been used by Drezner [44] to solve small size problems, it has further been studied by Resende et al. [112] who used an interior point algorithm to solve the resulting linear programs.

Lower bounds based on semi-definite formulation

A generalization of linear programming with variables taken from the Euclidean space of matrices is called semi-definite programming. In this case the non-negativity constraints are replaced by semi-definite constraints and linear constraints are formulated in terms of linear operators on the Euclidean space of matrices, the trace operator acting as an inner product. Due to its success (see [89],[57]), semi-definite programming regained interest in discrete optimization. Relaxations of the $\mathcal{QAP}$ using semi-definite programming have been studied by Karish [75], Zhao [143] and Zhao et al. [144] using interior point methods. Semi-definite programming relaxations dominate the relaxation of Adams and Johnson (**P 2.10**) as reported by Zhao et al. [144], but due to their high computation time requirement, it is not recommended to use them in branch and bound schemes.

Cutting plane methods

Other attempts to solve $\mathcal{QAP}$ using approaches that do not rely on branch and bound have been made. In this context Kaufman and Broeckx [77] proposed a cutting plane method based on the Bender's decomposition scheme applied to the linear transformation (2.6) of the $\mathcal{QAP}$ discussed above. The same scheme was applied by Bazaraa and Sherali [12, 13] to their linear transformation (**P 2.8**).

Balas and Mazzola [5] apply a different approach to the following transformation of the $\mathcal{QAP}$

P 2.11

$$\textit{minimize} \quad z$$

$$\textit{subject to} \quad :$$

$$\sum_{j=1}^{n}\sum_{l=1}^{n}\left(f_{jl}y_{jl}+\sum_{i=1}^{n}\sum_{k=1}^{n}a_{ij}b_{kl}y_{ik}\right)x_{jl}-\sum_{j=1}^{n}\sum_{l=1}^{n}f_{il}y_{jl}\leq z$$

$$\sum_{i=1}^{n}x_{ij}=1 \qquad \forall j=1...n$$

$$\sum_{j=1}^{n}x_{ij}=1 \qquad \forall i=1...n$$

$$\sum_{i=1}^{n}y_{ij}=1 \qquad \forall j=1...n$$

$$\sum_{j=1}^{n}y_{ij}=1 \qquad \forall i=1...n$$

$$x_{ij}\in\{0,1\},\ y_{ij}\in\{0,1\}$$

where

$$f_{jl}=\max\left\{\sum_{i=1}^{n}\sum_{k=1}^{n}a_{ij}b_{kl}y_{ik}:\sum_{i=1}^{n}y_{ij}=1,\ \sum_{j=1}^{n}y_{ij}=1,\ y_{ij}\in\{0,1\}\ i,j=1...n\right\}$$

For the nonlinear constraint an equivalent family of linear constraints is defined and the linear relaxation is solved to optimality. Linear inequalities corresponding to the constraints violated in the original problem are added to build a stronger linear relaxation. This process is repeated until a solution to the original problem is obtained. Some work using polyhedral cutting planes is described by Padberg and Rijal [99].

2.6.3 Heuristic algorithms

Construction methods

These are considered to be the simplest heuristic approaches to the $\mathcal{QAP}$. This simplicity is also associated with a poor quality solution. Mainly all construction methods start with an empty partial assignment and recursively assign facilities to locations based on some criteria until all facilities are assigned. Such heuristics have been proposed by Burkard and Startman [26], Conway and Maxwell [39], Edwards et al. [46], Gilmore [56], then refined by Burkard [19], Hillier and Connors [69], Heider [65, 67], Parker [103] and Wimmert [141]. Another construction method known as the method of increasing degrees of freedom is due to Muller-Merbach [97] will be discussed in chapters 6.

Limited enumeration

In many exact algorithms based on branch and bound approaches, it was found that optimal or close to optimal solutions are often found at an early stage of the search, the remaining large amount of time is spent to check optimality. This suggested methods, called *limited enumeration methods*, that try to take advantage of such information to derive good heuristic algorithms. Different approaches are used to limit the enumeration. Such approaches can be summarized as to whether, stop enumeration after a pre-specified amount of time, or rather, increase the lower bound at nodes of the tree that have not yet been visited if there is no improvement after

a pre-specified amount of time. Limited enumeration has also been applied to speed up cutting plane algorithms, by limiting the number of cuts used.
Such approaches have been used by Bazaraa and Sherali [12] and Burkard and Bonniger [21].

Improvement methods

These are classical approaches to solve difficult combinatorial optimization problems. They start with an initial feasible solution then iteratively try to improve the current solution by pairwise exchange or a cycle triple exchange. In the pair exchange, the neighborhood of the current solution is scanned, it consists of all pair exchanges of indices. Heuristics in this class were proposed by Armour and Buffa [3], Bazaraa and Elshafei [10], Heider [66], Hillier [68], Hillier and Connors [69], Nugent et al. [98], Patel et al. [104], Pegels [105] and Vollmann et al. [133]. In the case of cycle triple exchanges, the neighborhood of a feasible solution is scanned by checking all cycle exchange of three indices. If three indices i, j, k from a permutation π are to be cycle exchanged, we obtain two permutations π' and π'' such that :

$$\pi'(x) = \begin{cases} \pi(x) \text{ if } x \neq (i,j,k) \\ \pi(k) \text{ if } x = i \\ \pi(i) \text{ if } x = j \\ \pi(j) \text{ if } x = k \end{cases}$$

and

$$\pi''(x) = \begin{cases} \pi(x) \text{ if } x \neq (i,j,k) \\ \pi(j) \text{ if } x = i \\ \pi(k) \text{ if } x = j \\ \pi(i) \text{ if } x = k \end{cases}$$

Such strategy was investigated by Brujis [16] with different updating rules. Combining pair-exchange and cyclic triple exchange strategies have been suggested by Mirchandani and Obata [93]

2.6.4 Meta-heuristics

Meta-heuristics such as Tabu search, Simulated Annealing, and Genetic Algorithms turned out to be very efficient in approximating the optimal solution, over stepping the first heuristics solution methods (construction methods, exchange methods). Simulated Annealing based algorithms were first applied to $\mathcal{QAP}$ by Burkard and Rendl [25]. A more sophisticated algorithm was later proposed by Wilhelm and Ward [140] and then by Laursen [85]. Genetic Algorithms for the $\mathcal{QAP}$ were developed by Tate and Smith [129], then improved by Fleurent and Feland [50] who combine genetic algorithms with other heuristic procedures for the $\mathcal{QAP}$ in a so-called hybrid approach. Ahuja et al. [2] obtained very promising results for large scale $\mathcal{QAP}$ by applying a version of Genetic Algorithms they call *greedy genetic algorithm*. Tabu search was used by Skorin-Kapov [124, 125] then improved by Taillard [128] in a so-called *robust tabu search*. Recently, another version of tabu search called

reactive tabu search was proposed by Battiti and Tecchiolli [8] with very good results. Other evolutionary approaches have also been applied to the $\mathcal{QAP}$, in this context we find an algorithm based on scatter search proposed by Cung et al. [41], it uses linear combinations of a population subset to create new solutions. Neural network approaches were also studied in the case of the $\mathcal{QAP}$, such approach was used by Tsuchiya et al. [131].

2.6.5 Parallel algorithms

Now and due to the development of parallel architecture, many parallel algorithms based on Branch and Bound have been proposed. From these we have Bruenegger et al. [17], Clausen and Perregaard [37], Laursen [84], Pardalos and Crouse [100], Roucairol [114], Crouse and Pardalos [40] but even those were limited to small sizes. In 1995 Mans et al. [91] proposed a parallel branch and bound algorithm using a depth first search and claim that it provides better results in terms of number of nodes that need to be investigated. Heuristics that worked well for the $\mathcal{QAP}$ in the sequential case have also been used with parallel implementation, in this context Pardalos et al. [101], provide a parallel implementation of $\mathcal{GRASP}$ (Greedy Randomized Adaptive Search). Also as a consequence of the good behavior of the Meta-heuristics (Tabu search, Simulated Annealing, and Genetic Algorithms) in the sequential case and to the fact that these are highly parallelizable, parallel implementation have been proposed Brown et al. [17], have developed a parallel Genetic Algorithm on a 32-

nodes Hyper-cube, during the same year another parallel Genetic Algorithm on a 64 processors system with distributed memory was developed by Mühlenbein[96]. A parallel implementation of the so-called *genetic algorithms with local search* is given by Huntley and Brown [71].

In 1991 parallel algorithms based on Tabu search were concurrently developed by Taillard [128] on a network of 10 transputers, and Chakrapani and Skorin-Kapov [30] on a Connection Machine using $16K$ processing units. The optimal solution was always found for small problems, but for larger problems optimality can not be proved due to the lack of solved problems with large size, although for problems from the literature [98] built with known optimal solution the results of these heuristics were very close to the optimal solution.

2.7 Problems related to the $\mathcal{QAP}$

Many well known combinatorial optimization problem can be considered as related to the $\mathcal{QAP}$ in the sense that they represent a generalization or specialization of the $\mathcal{QAP}$.

May be the first one that comes to mind is the linear assignment problem ($\mathcal{LAP}$). In this problem, the interaction cost, that is at the origin of the difficulty encountered with the $\mathcal{QAP}$ is not taken into consideration, hence leading to an easy problem with many known efficient algorithms.

Other more difficult problems related to the $\mathcal{QAP}$ are given bellow.

2.7.1 The quadratic set covering problem:

The quadratic set covering problem or $\mathcal{QSCP}$ can be defined as follows: Let X be an n vector, Q be an $n \times n$ matrix, and A an $m \times n$ matrix of 0 and $1's$, the $\mathcal{QSCP}$ is formulated as follows:

$$QSCP \begin{cases} \min X^t Q X \\ AX \geq e_m \\ X \text{ binary} \end{cases}$$

where e_m represents a vector of m ones.

The problem attempts to find the minimum cost for covering m markets from n warehouses. More than one warehouse may cover the same market, as long as all markets are covered. The entries of the matrix $A = \{a_{ij}\}_{j=1\ldots n}^{i=1,\ldots,m}$ are binary such that:

$$a_{ij} = \begin{cases} 1 \text{ if warehouse } i \text{ can covers market } j \\ 0 \text{ Otherwise} \end{cases}$$

2.7.2 The quadratic set partitioning problem:

A special case of $\mathcal{QSCP}$ is the quadratic set partitioning problem or $\mathcal{QSPP}$. In this case each market is required to be covered by exactly one warehouse. It is formulated as follows:

$$QSPP \begin{cases} \min X^t Q X \\ AX = e_m \\ X \text{ binary} \end{cases}$$

2.7.3 The traveling salesman problem:

The traveling salesman problem or $\mathcal{TSP}$, a very well known problem to operations researchers, is another example of problems related to the $\mathcal{QAP}$. It has the same formulation as the $\mathcal{QAP}$, the only difference is that the flow matrix in this case has a special structure. In the case of $\mathcal{TSP}$, we are given n cities and a matrix D representing the distance between cities. The objective is to find and order to travel through all the cities and back to the initial city with a minimum traveling distance, or minimum traveling cost in case where D represents a cost instead of the distances. The problem is formulated as follows:

$$TSP \begin{cases} \textit{minimize } Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} f_{il}d_{jm}x_{ij}x_{lm} \\ \textit{subject to :} \\ \qquad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1...n \\ \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1...n \\ \qquad x_{ij} \in \{0,1\} \end{cases}$$

Here for $i, l = 1...n$ we define f_{il} to be such that, $f_{il} = \begin{cases} 1 & \text{if } l = i+1 \text{ and } i < n \\ 1 & \text{if } i = n \text{ and } l = 1 \\ 0 & \text{otherwise} \end{cases}$

and x_{ij}, $i, j = 1...n$ is defined as $x_{ij} = \begin{cases} 1 & \text{if city } j \text{ is visited after city } i \\ 0 & \text{otherwise} \end{cases}$

2.8 Conclusion

The above review shows that the $\mathcal{QAP}$ is a challenging problem that attracted many researchers. It is very much an active area of research. In the next chapters, the contribution made in this dissertation are presented.

Chapter 3

The $\mathcal{QAP}$ on the graph

3.1 Introduction

The purpose of this chapter is to exploit the network structure of the $\mathcal{QAP}$ feasible set and characterize a local minimum for the $\mathcal{QAP}$. The network structure then used to propose various ways to develop algorithms for solving the $\mathcal{QAP}$.

It is well known that a basic feasible solution for a network model can be represented as a spanning tree and that an entering arc will provoke the creation of a cycle, therefore another arc has to leave the basis in order to obtain a new basic feasible solution. In what follows, when we talk about a spanning tree, we mean the spanning tree corresponding to a basic feasible solution in the network, and when we refer to a cycle we mean the cycle created after adding a new non-basic arc to the spanning tree. Given a basic feasible solution, if we introduce a non-basic variable (non-basic

arc) and increase the flow on the corresponding arc by a value $\delta > 0$, a cycle is created (due to adding this non-basic variable), consequently the flow on some arcs of the cycle will decrease while on others it will increase. This happens in order to maintain feasibility.

This pivot is going to be explored in the case of the $\mathcal{QAP}$ and will help providing rules on how to update a basis. These rules are then used to suggest an extreme point based algorithm for the $\mathcal{QAP}$.

In the next section, the notation used is defined followed by the introduction of a mapping that is used to characterize the extreme points. Some properties of the mapping are also discussed.

The extreme points of the feasible set of the assignment polytope are discussed in section 3.2.

In section 3.3 results concerning the characterization of a local minimum of the $\mathcal{QAP}$ with respect to a given basis are presented and proved, an example is discussed to show the use of the characterization.

In the following section, an algorithm using the results in section 3.3 is proposed.

Section 3.5 concludes the chapter

3.2 Notation

Given a basic feasible solution for the quadratic assignment problem, the following notation will be used throughout the whole chapter.

$\Re^+$ The set of positive real numbers

A_G The set of all arcs in the graph

$\mathcal{U}$ The power set of A_G, i.e. the set of all subsets of A_G

B The set of basic arcs (i.e. arcs on the spanning tree of the basic solution)

B_+ The set of basic arcs with positive flow (active arcs)

B_0 The set of basic arcs with zero flow (inactive arcs)

B_+^δ The active arcs of the spanning tree whose flow decreases by a value $\delta > 0$ upon increasing the flow on a given non-basic arc

B_0^δ The inactive arcs of the spanning tree whose flow increases by a value $\delta > 0$ upon increasing the flow on a given non-basic arc

B_+^1 The set of arcs in $B_+ \setminus B_+^\delta$

B_0^0 The set of arcs in $B_0 \setminus B_0^\delta$.

Let us define the mapping

$$\mathcal{C} : \mathcal{U} \times \mathcal{U} \longrightarrow \Re^+ \cup \{0\}$$

The mapping $\mathcal{C}$ is intended to correspond to the interaction cost of the quadratic function in the $\mathcal{QAP}$. Then given any two arcs (i,j) and (k,l) we have:

$$\mathcal{C}(\{(i,j)\},\{(k,l)\}) = \frac{1}{2}(f_{ik}d_{jl} + f_{ki}d_{lj}) \tag{3.1}$$

In the same order, given two sets $S_1 \subseteq A_G$ and $S_2 \subseteq A_G$ we define:

$$\mathcal{C}(S_1,S_2) = \sum_{\substack{[(i,j),(k,l)]\in S_1\times S_2 \\ i\neq k \text{ and } j\neq l}} \mathcal{C}(\{(i,j)\},\{(k,l)\}) \tag{3.2}$$

The above mapping $\mathcal{C}$ has the following properties:

Properties : Given three mutually exclusive sets X, Y and Z, and an arc (i,j)

$$1- \quad \mathcal{C}(\{(i,j)\},\{(i,j)\}) = 0$$

$$2- \quad \mathcal{C}(X,Y) = \mathcal{C}(Y,X)$$

$$3- \quad \mathcal{C}(X,\{(i,j)\}) = \tfrac{1}{2}\sum_{\substack{(k,l)\in X \\ k\neq i, l\neq j}} (f_{ik}d_{jl} + f_{ki}d_{lj})$$

$$4- \quad \mathcal{C}(X\cup Y, Z) = \mathcal{C}(X,Z) + \mathcal{C}(Y,Z)$$

$$5- \quad \mathcal{C}(X, Y\cup Z) = \mathcal{C}(X,Y) + \mathcal{C}(X,Z)$$

$$6- \quad \mathcal{C}(X\cup Y, X\cup Y) = \mathcal{C}(X,X) + \mathcal{C}(Y,Y) + 2\mathcal{C}(X,Y)$$

Proof.

Property 1 follow directly from the fact that $d_{jj} = 0$ and $f_{ii} = 0$.

Property 2 is a direct consequence of the commutative property of the multiplication and addition.

Property 3 is straightforward, since X consists of a set of arcs (k,l) that can be classified as follows: X_1 consisting of those arcs in X not incident to any of nodes i or j and X_2 consisting of arcs in X incident to nodes i or j or both (i.e. arcs of the form $(i,l),(k,j)$ or (i,j) itself, where $k \neq i$ and $j \neq l$), for the set X_1 it is clear that $\mathcal{C}(X_1,\{(i,j)\}) = \frac{1}{2}\sum_{(k,l)\in X_1}(f_{ik}d_{jl} + f_{ki}d_{lj})$. For the set X_2, because $d_{jj} = 0$ and $f_{ii} = 0$ then $\mathcal{C}(X_2,\{(i,j)\}) = 0$.

Properties 4 and 5 are direct application of the mapping $\mathcal{C}$ taking into consideration that the three sets X, Y and Z are mutually exclusive.

Property 6 follows from properties 4 and 5 as follows:

$$\begin{aligned} C(X \cup Y, X \cup Y) &= C(X, X \cup Y) + C(Y, X \cup Y) \\ &= C(X, X) + C(X, Y) + C(Y, X) + C(Y, Y) \\ &= C(X, X) + 2C(X, Y) + C(Y, Y) \end{aligned}$$

Since $C(X, Y) = C(Y, X)$. ■

Hence the value of the objective function for the basic solution B is given by:

$$C(B, B) = C(B_+, B_+) \tag{3.3}$$

This is true because $C(B_+, B_0)$ and $C(B_0, B_0)$, will be multiplied by x_{ij} where $(i, j) \in B_0$, and $x_{ij} = 0$.

3.3 Extreme points of the $\mathcal{QAP}$.

We will start this section by stating a theorem that is fundamental for the rest of the chapter

Theorem 3 *Every feasible solution of the constraints of problem (1.2) is an extreme point of the linear relaxation of the constraints.*

Proof.

Suppose that there exists a feasible solution x^0 that is not an extreme point, then there exists a set of distinct extreme points $x^1, x^2, ..., x^k$, $x^0 \neq x^l$ $l = 1, ..., k$ such that:

$$x^0 = \lambda_1 x^1 + \lambda_2 x^2 + ... + \lambda_k x^k \quad \text{where} \quad \sum_{i=1}^{k} \lambda_i = 1 \text{ and } \lambda_i > 0, \ \forall i = 1, .., k \qquad (3.4)$$

Hence every component α_{ij}^0 of x^0 is of the form:

$$\alpha_{ij}^0 = \lambda_1 \alpha_{ij}^1 + \lambda_2 \alpha_{\alpha ij}^2 + ... + \lambda_k \alpha_{ij}^k \qquad 0 < \lambda_i < 1, \ \forall i = 1, .., k \qquad (3.5)$$

By feasibility of x^0, $\alpha_{ij}^0 \in \{0, 1\}$.

If $\alpha_{ij}^0 = 0$, then $\lambda_1 \alpha_{ij}^1 + \lambda_2 \alpha_{ij}^2 + ... + \lambda_k \alpha_{ij}^k = 0 \Rightarrow \alpha_{ij}^1 = \alpha_{ij}^2 = ... = \alpha_{ij}^k = 0$ this is because $0 < \lambda_i < 1$, $\forall i = 1, .., k$ and α_{ij}^1, $\alpha_{ij}^2, ..., \alpha_{ij}^k$ are all non-negative (in fact α_{ij}^1, $\alpha_{ij}^2, ..., \alpha_{ij}^k \in \{0, 1\}$).

On the other hand, if $\alpha_{ij}^0 = 1$, then $\lambda_1 \alpha_{ij}^1 + \lambda_2 \alpha_{ij}^2 + ... + \lambda_k \alpha_{ij}^k = 1$. For this equality to hold given that $0 < \lambda_i < 1$, $\forall i = 1, .., k$, we must have $\alpha_{ij}^1 = \alpha_{ij}^2 = ... = \alpha_{ij}^k = 1$ due to the fact that $\sum_{i=1}^{k} \lambda_i = 1$.

Hence $\alpha_{ij}^0 = \alpha_{ij}^1 = \alpha_{ij}^2 = ... = \alpha_{ij}^k$, $\forall i, j$ which means that $x^0 = x^1 = x^2 = ... = x^k$. This contradicts the fact that x^1, $x^2, ..., x^k$ are distinct. Hence there is no feasible solution that is not an extreme point. ■

From the above theorem, it is clear that any local or global solution to the $\mathcal{QAP}$ has to be an extreme point. Having this in mind, we can solve the problem if we can have at hand a good scheme that permits us to characterize a *local minimum* and move around extreme points. A parallel view of the simplex method on the network gives us a flavor of how to obtain this scheme.

In the linear case, the arc costs c_{ij} play a major role in deciding on which variable to enter the basis. These c_{ij} are nothing but the components of the gradient of the cost function with respect to the variable x_{ij}. For the $\mathcal{QAP}$, this luxury is not available, but one can consider the gradient of the quadratic function at an extreme point and use it as an alternative to the arc cost c_{ij}, this may help us to decide on which extreme point to move to in order to improve the current solution at hand.

3.4 Characterization of a B-local star minimum

In this section, the above mentioned idea is explored for characterizing a relative local star minimum [B-local star minimum], that we define next. The following definitions are made with respect to a given $\mathcal{QAP}$ with an objective function f.

Definition 13 *A local star minimum x^0 is an extreme point that has an objective function value less than or equal to any of its adjacent extreme point. i.e.*

$$f(x^0) \leq f(x^j) \qquad\qquad j = 1, ..., p$$

where x^j is an adjacent extreme point of x^0 and p is the number of adjacent extreme points.

Next , we provide a definition of a relative local star minimum, since an extreme point in the case of an assignment polytope can be represented by more than one basis as given in section 2.2.

Definition 14 *A relative local star minimum [B-local star minimum] is a point satisfying the local star minimum condition with respect to a given basis (represented by the spanning tree B).*

Consider an extreme point x^0, and the objective function value at that point Z^0. Clearly the basic variables x_{ij}^0 of x^0 can be viewed as a set of arcs in the spanning tree B. Using the notation of the mapping defined above, Z^0 will have the following form:

$$\begin{aligned} Z^0 &= C(B, B) \\ &= C(B_+, B_+) \end{aligned}$$

Clearly, x^0 corresponds to a spanning tree, in the bipartite graph representing the assignment problem at hand, and a simplex pivot implies the formation of a cycle by entering an arc say (k, l). The flow on this arc can be taken to be $\delta > 0$, we consider here the relaxed assignment problem (when the integrality constraints are relaxed or $0 \leq x_{ij} \leq 1$). This implies a rising in the flow on some arcs in B_0 (those inactive arcs appearing on the cycle) to a value of δ, The set of such arcs is called B_0^δ. On the other hand the flow will decrease by a value δ on some arcs in B_+ (those active arcs appearing on the cycle), the set of such arcs is called B_+^δ . Given the above notation for a basic feasible solution, it is clear that $B_+ = B_+^1 \cup B_+^\delta$, $B_0 = B_0^0 \cup B_0^\delta$ and $B = B_+ \cup B_0$. Using the fact that $B_+ = B_+^1 \cup B_+^\delta$, and applying property 4 of the

mapping $\mathcal{C}$, we can see that :

$$\begin{aligned} Z^0 &= \mathcal{C}(B_+, B_+) \\ &= \mathcal{C}(B_+^1 \cup B_+^\delta, B_+^1 \cup B_+^\delta) \\ &= \mathcal{C}(B_+^1, B_+^1) + 2\mathcal{C}(B_+^1, B_+^\delta) + \mathcal{C}(B_+^\delta, B_+^\delta) \end{aligned}$$

Hence Z^δ, the value of the objective function when adding the arc (k, l) with a flow δ and relaxing the integrality condition, will have the following form:

$$\begin{aligned} Z^\delta &= \mathcal{C}(B \cup \{(k,l)\}, B \cup \{(k,l)\}) \\ &= \mathcal{C}(B_+^1 \cup B_+^\delta \cup B_0^\delta \cup \{(k,l)\}, B_+^1 \cup B_+^\delta \cup B_0^\delta \cup \{(k,l)\}) \\ &= \mathcal{C}(B_+^1, B_+^1) + (1-\delta)\mathcal{C}(B_+^1, B_+^\delta) + \delta\mathcal{C}(B_+^1, B_0^\delta) + \delta\mathcal{C}(B_+^1, \{(k,l)\}) \\ &\quad +(1-\delta)\mathcal{C}(B_+^\delta, B_+^1) + (1-\delta)^2\mathcal{C}(B_+^\delta, B_+^\delta) + \delta(1-\delta)\mathcal{C}(B_+^\delta, B_0^\delta) \\ &\quad +\delta(1-\delta)\mathcal{C}(B_+^\delta, \{(k,l)\}) + \delta\mathcal{C}(B_0^\delta, B_+^1) + \delta(1-\delta)\mathcal{C}(B_0^\delta, B_+^\delta) \\ &\quad +\delta^2\mathcal{C}(B_0^\delta, B_0^\delta) + \delta^2\mathcal{C}(B_0^\delta, \{(k,l)\}) + \delta\mathcal{C}(\{(k,l), B_+^1\}) \\ &\quad +\delta(1-\delta)\mathcal{C}(\{(k,l)\}, B_+^\delta) + \delta^2\mathcal{C}(\{(k,l)\}, B_0^\delta) + \delta^2\mathcal{C}(\{(k,l)\}, \{(k,l)\}) \end{aligned}$$

Because $\mathcal{C}(\{(k,l)\}, \{(k,l)\}) = 0$, and using property 2, we get

$$\begin{aligned} Z^\delta &= [\mathcal{C}(B_+^1, B_+^1) + 2\mathcal{C}(B_+^1, B_+^\delta) + \mathcal{C}(B_+^\delta, B_+^\delta)] + \delta[2\mathcal{C}(B_+^1, B_0^\delta) \\ &\quad +2\mathcal{C}(B_+^\delta, B_0^\delta) + 2\mathcal{C}(B_+^1, \{(k,l)\}) + 2\mathcal{C}(B_+^\delta, \{(k,l)\}) - 2\mathcal{C}(B_+^1, B_+^\delta) - 2\mathcal{C}(B_+^\delta, B_+^\delta)] \\ &\quad +\delta^2[\mathcal{C}(B_+^\delta, B_+^\delta) + \mathcal{C}(B_0^\delta, B_0^\delta) + 2\mathcal{C}(B_0^\delta, \{(k,l)\}) - 2\mathcal{C}(B_+^\delta, B_0^\delta) - 2\mathcal{C}(B_+^\delta, \{(k,l)\})] \end{aligned}$$

Hence Z^δ can be written as quadratic form as follows:

$$Z^\delta = a\delta^2 + b\delta + c \tag{3.6}$$

Proposition 1 *In the quadratic form (3.6) when a non-basic arc (k,l) is considered for pivoting, it can be seen that:*

(i) $Z^\delta\big|_{\delta=0} = c = Z^0$

(ii) $\frac{\partial Z^\delta}{\partial \delta}\big|_{\delta=0} = b$

Proof.

(i) $Z^\delta\big|_{\delta=0} = c = Z^0$:

Using the fact that $B_+ = B_+^1 \cup B_+^\delta$, and applying property 4 of the mapping $\mathcal{C}$, we can see that :

$$\begin{aligned} Z^0 &= \mathcal{C}(B_+, B_+) \\ &= \mathcal{C}(B_+^1 \cup B_+^\delta, B_+^1 \cup B_+^\delta) \\ &= \mathcal{C}(B_+^1, B_+^1) + 2\mathcal{C}(B_+^1, B_+^\delta) + \mathcal{C}(B_+^\delta, B_+^\delta) \end{aligned}$$

and replacing δ by 0 in the expression of Z^δ given below

$$\begin{aligned} Z^\delta &= [\mathcal{C}(B_+^1, B_+^1) + 2\mathcal{C}(B_+^1, B_+^\delta) + \mathcal{C}(B_+^\delta, B_+^\delta)] + \delta[2\mathcal{C}(B_+^1, B_+^\delta) \\ &\quad +2\mathcal{C}(B_+^\delta, B_0^\delta) + 2\mathcal{C}(B_+^1, \{(k,l)\}) + 2\mathcal{C}(B_+^\delta, \{(k,l)\}) - 2\mathcal{C}(B_+^1, B_+^\delta) - 2\mathcal{C}(B_+^\delta, B_+^\delta)] \\ &\quad +\delta^2[\mathcal{C}(B_+^\delta, B_+^\delta) + \mathcal{C}(B_0^\delta, B_0^\delta) + 2\mathcal{C}(B_0^\delta, \{(k,l)\}) - 2\mathcal{C}(B_+^\delta, B_0^\delta) - 2\mathcal{C}(B_+^\delta, \{(k,l)\})] \end{aligned}$$

We get, $Z^{\delta}\big|_{\delta=0} = C(B^1_+, B^1_+) + 2C(B^1_+, B^{\delta}_+) + C(B^{\delta}_+, B^{\delta}_+) = Z^0$, this is nothing but c the constant coefficient of Z^{δ} in (3.6).

(ii) $\frac{\partial Z^{\delta}}{\partial \delta}\big|_{\delta=0} = b$

Clearly, the component of the gradient with respect to any variable x_{ij} is given by:

$$g_{ij} = \frac{\partial f}{\partial x_{ij}}(X) = \sum_{\substack{k=1 \\ k\neq i}}^{n} \sum_{\substack{l=1 \\ l\neq j}}^{n} (f_{ik}d_{jl} + f_{ki}d_{lj})x_{kl} \tag{3.7}$$

This is obtained directly from the objective function. If we now evaluate the gradient at any extreme point x^0, the term $(f_{ik}d_{jl} + f_{ki}d_{lj})$ will appear in the evaluation of the gradient only if $x_{kl} \in B_+$ (otherwise it will be multiplied by $x_{kl} = 0$). Hence, we have:

$$\begin{aligned} g_{ij} &= \frac{\partial f}{\partial x_{ij}}(X^0) = \sum_{\substack{(k,l)\in B_+ \\ k\neq i, l\neq j}} (f_{ik}d_{jl} + f_{ki}d_{lj}) \\ &= 2C(B_+, \{(k,l)\}) \end{aligned}$$

Now, let us consider the spanning tree representing the extreme point x^0, and the cycle obtained by entering the arc (k,l) with a flow $\delta > 0$. Using the above notation, the cycle is composed of arcs in $B^{\delta}_+ \cup B^{\delta}_0 \cup \{(k,l)\}$ as shown below:

Then performing the simplex pivot on the arc (k,l) with g_{ij} playing the same role as the costs in the linear case, we first evaluate the dual variables w_i's for the

nodes adjacent to arcs in $B_+^\delta \cup B_0^\delta$. It is clear that arcs (k,j) and (i,l) belong to B_+^δ. Starting with

$$w_k = 0$$

and using the spanning tree representing the basis, we get:

$$w_j = -g_{kj}$$

$$w_m = g_{mj} - g_{kj}$$

.

.

.

$$w_l = \sum_{(i,j)\in B_0^\delta} g_{ij} - \sum_{(i,j)\in B_+^\delta} g_{ij}$$

Then we get what we call the " *reduced gradient* " at x_{kl} that is given by:

$$\begin{aligned} z_{kl} - g_{kl} &= w_k - w_l - g_{kl} \\ &= \sum_{(i,j)\in B_+^\delta} g_{ij} - \sum_{(i,j)\in B_0^\delta} g_{ij} - g_{kl} \\ &= 2\mathcal{C}(B_+, B_+^\delta) - 2\mathcal{C}(B_+, B_0^\delta) - 2\mathcal{C}(B_+, \{(k,l)\}) \end{aligned}$$

But from the properties of the mapping $\mathcal{C}$, we have:

$$\begin{aligned} \mathcal{C}(B_+, B_+^\delta) &= \mathcal{C}(B_+^1, B_+^\delta) + \mathcal{C}(B_+^\delta, B_+^\delta) \\ \mathcal{C}(B_+, B_0^\delta) &= \mathcal{C}(B_+^1, B_0^\delta) + \mathcal{C}(B_+^\delta, B_0^\delta) \\ \mathcal{C}(B_+, \{(k,l)\}) &= \mathcal{C}(B_+^1, \{(k,l)\}) + \mathcal{C}(B_+^\delta, \{(k,l)\}) \end{aligned}$$

Plugging in the expression of $z_{kl} - g_{kl}$ we get:

$$\begin{aligned} z_{kl} - g_{kl} &= 2\mathcal{C}(B^1_+, B^\delta_+) + 2\mathcal{C}(B^\delta_+, B^\delta_+) \\ &\quad -2\mathcal{C}(B^1_+, B^\delta_0) - 2\mathcal{C}(B^\delta_+, B^\delta_0) \\ &\quad -2\mathcal{C}(B^1_+, \{(k,l)\}) - 2\mathcal{C}(B^\delta_+, \{(k,l)\}) \\ &= -b \end{aligned}$$

Hence $-b$ is what we called previously the reduced gradient at x_{kl}. ■

Remark 1 *In the above proposition, we do not need to compute Z^δ, but rather, we can easily compute its coefficients and use the following theorem to decide whether to perform a pivot or not. The way $-b$ is computed is explained latter in the example.*

On way to compute a without computing Z^δ is explained in the following remark

Remark 2 *If we define the cost of combining two arcs say (i,j) and (k,l) as the value of the mapping $\mathcal{C}$ at $(\{(i,j)\}, \{(k,l)\})$, [i.e. $\mathcal{C}(\{(i,j)\}, \{(k,l)\})$]. The coefficient a of δ^2, would easily be computed by taking the sum of the cost of combining arcs in B^δ_+ together, and arcs in B^δ_0 together, and subtracting from it the cost of combining arcs in B^δ_+ with arcs in B^δ_0.*

Theorem 4 *The extreme point x^0 is a relative local star minimum if one of the following is satisfied:*

$$\begin{cases} i & a > 0 \text{ and } -\frac{b}{a} < 1, \forall (i,j) \in A, \text{ or} \\ ii & a < 0 \text{ and } -\frac{b}{a} > 1 \ \forall (i,j) \in A. \end{cases}$$

Where a and b represent the coefficient of δ^2 and δ respectively in the expression of Z^δ .

Proof. $Z^\delta - Z^0 = a\delta^2 + b\delta = \delta[a\delta + b]$,

If $a > 0$, then an improvement in the objective function needs $Z^\delta - Z^0 < 0 \Rightarrow a\delta^2 + b\delta < 0 \Rightarrow \delta[a\delta + b] < 0$ since $\delta > 0$, the last inequality is equivalent to $a\delta + b < 0 \Rightarrow a\delta < -b \Rightarrow \delta < -\frac{b}{a}$. By assumption, $\delta < -\frac{b}{a} < 1$. Hence, we cannot increase δ to 1 meaning that x^0 is a relative local star minimum.

If $a < 0$, then an improvement in the objective function means $Z^\delta - Z^0 < 0 \Rightarrow a\delta^2 + b\delta < 0 \Rightarrow \delta[a\delta + b] < 0$ since $\delta > 0$, the last inequality is equivalent to $a\delta + b < 0 \Rightarrow a\delta < -b \Rightarrow \delta > -\frac{b}{a}$ [since $a < 0$]. By assumption, $-\frac{b}{a} > 1$. Hence, to improve, we must raise δ beyond 1, which is impossible since $\delta \in [0, 1]$, this means that x^0 is a relative local star minimum. ■

Theorem 5 *Given an extreme point x^0. The extreme point obtained by entering a non-basic variable x_{kl} into the basis is improving if and only if $a < -b$. Where b and a are the coefficients of δ and δ^2 respectively in the expression of Z^δ. The value of the improvement in such a case is equal to $a + b$.*

Proof. $Z^\delta - Z^0 = a\delta^2 + b\delta = \delta[a\delta + b]$. Since there exists x_{kl} for which the ratio $\frac{b}{a}$ satisfies the condition for pivoting (see theorem 3.1), we can make the flow on the entering arc δ to be 1, which satisfies the integrality conditions of the solution, and for

which $Z^{\delta} - Z^0 = a + b < 0$, this provides us with a new extreme point that improves the value of the objective function. Clearly the value of the improvement is $a + b$. ■

In what follows, we give a detailed example:

Example 1 *Consider the QAP given by the assignment graph in* ***Figure*** 3.1 *and the associated flow and distance matrices.*

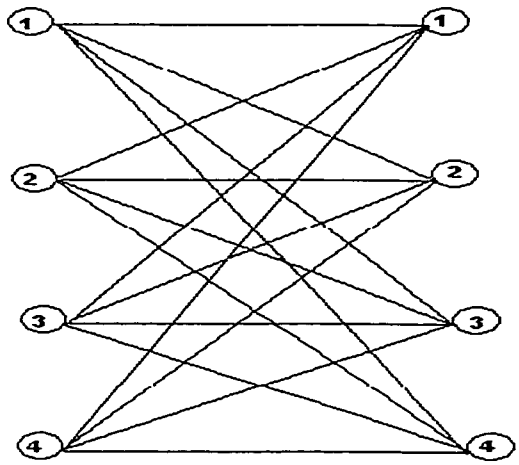

Figure 3.1: The Assignment Problem

$$f = \begin{vmatrix} 0 & 4 & 3 & 5 \\ 4 & 0 & 3 & 2 \\ 3 & 3 & 0 & 2 \\ 5 & 2 & 2 & 0 \end{vmatrix} \qquad d = \begin{vmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 5 & 3 \\ 3 & 5 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{vmatrix}$$

Suppose we start with the feasible assignment X^0 given by:

$$x_{ij} = \begin{cases} 1 & i = j \\ & \qquad i, j = 1, ..., 4 \\ 0 & otherwise \end{cases}$$

clearly, we can augment this assignment with arcs $(1,2), (2,3)$ and $(3,4)$ to obtain a spanning tree as shown on **Figure** 3.2.

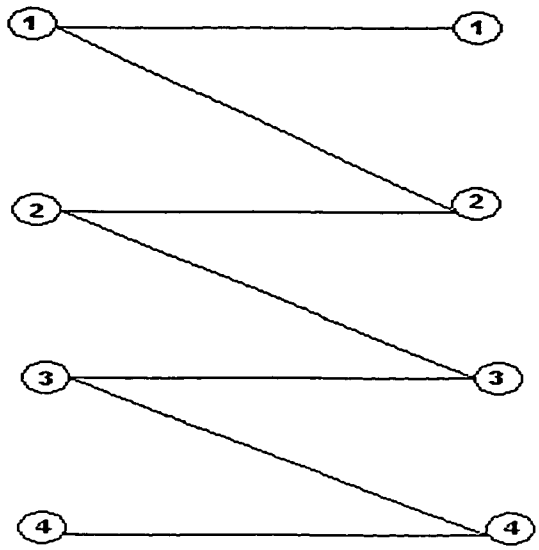

Figure 3.2: A spanning tree for the basic feasible solution

Computing the objective function value at x^0, we get: $Z = 104$ the gradient of the objective function at x^0 is given by:

$$\nabla f = \begin{vmatrix} 46 & 60 & 80 & 48 \\ 26 & 50 & 40 & 40 \\ 14 & 18 & 64 & 30 \\ 16 & 30 & 50 & 48 \end{vmatrix}$$

using this to compute the dual variables, where the g_{ij} (the components of the gradient) play the role of the cost coefficients in the linear case, we obtain the values in **bold** on **Figure** 3.2.

Using these dual variables, we can compute the negatives of the reduced gradients for the non-basic variables on the graph to get:

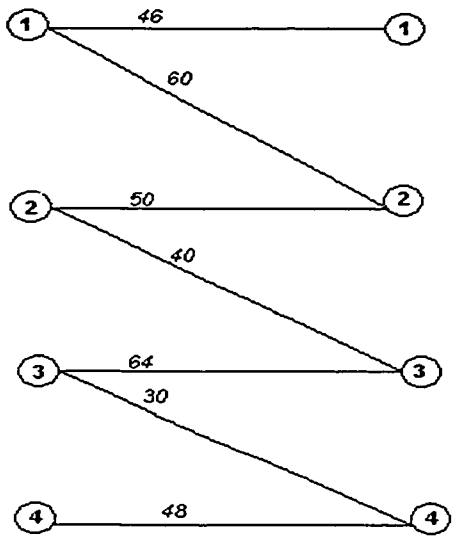

Figure 3.3: Reduced gradients for the non-basic variables

Remark 3 *The reduced gradients for the non-basic variables can easily be computed on an $n \times n$ table. Starting with the matrix whose entry (i, j) represents the partial derivative with respect to x_{ij} (see example above), we create zeros in the entries corresponding to basic variables the same way it is done when applying the simplex method to the transportation problem. This is illustrated for the above example in what follows:*

$$\nabla f = \begin{vmatrix} 46 & 60 & 80 & 48 \\ 26 & 50 & 40 & 40 \\ 14 & 18 & 64 & 30 \\ 16 & 30 & 50 & 48 \end{vmatrix} \rightarrow \begin{vmatrix} 0 & 14 & 34 & 2 \\ 26 & 50 & 40 & 40 \\ 14 & 18 & 64 & 30 \\ 16 & 30 & 50 & 48 \end{vmatrix} \rightarrow \begin{vmatrix} 0 & 0 & 34 & 2 \\ 26 & 36 & 40 & 40 \\ 14 & 4 & 64 & 30 \\ 16 & 16 & 50 & 48 \end{vmatrix} \rightarrow \ldots$$

Until we reach at end the following table:

$$\begin{vmatrix} 0 & 0 & 30 & 32 \\ -10 & 0 & 0 & 34 \\ -46 & -56 & 0 & 0 \\ -62 & -62 & -32 & 0 \end{vmatrix}$$

which corresponds to the reduced gradients for the non-basic variables.

And so, we see that variables x_{41} and x_{42} are candidates for entering the basis (those with most negative values). Choosing x_{42} to enter the basis, we have the formation of a cycle $(4,4)$, $(3,4)$, $(3,3)$, $(2,3)$, $(2,2)$ and $(4,2)$ as shown on **Figure** 3.4 below.

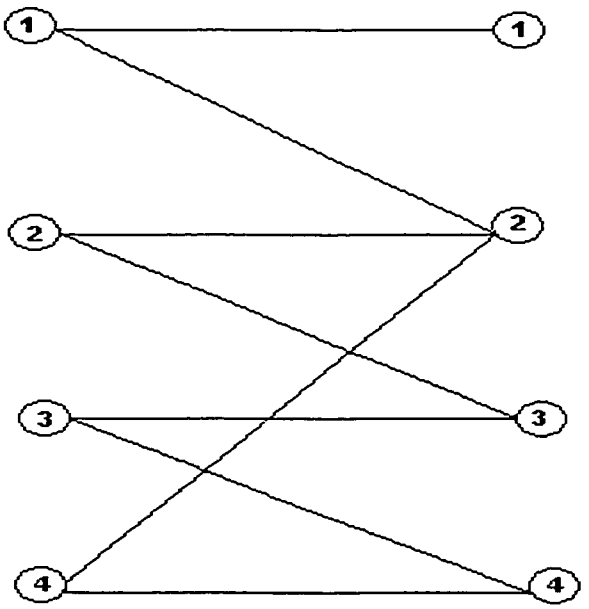

Figure 3.4: A cycle obtained after entering a non-basic variable

Raising the flow on $(4,2)$ by δ and computing Z^δ we get: $Z^\delta = 104 - 62\delta + 60\delta^2$. Since $-\frac{b}{a} = \frac{62}{48} > 1$ then we can improve the solution by entering x_{42} in the basis and raising its flow to 1, which forces x_{22} to leave the basis, leading to a new basic feasible solution given by:

$$x_{11} = x_{23} = x_{34} = x_{42} = 1$$

$$x_{33} = x_{44} = x_{12} = 0$$

with a new objective function value $Z = 102$.

Remark 4 *If we choose x_{41} to enter the basis, we get $Z^\delta = 104 - 62\delta + 70\delta^2$ then $-\frac{b}{a} = \frac{62}{70} < 1$, hence we cannot improve the solution as in that case the new $Z = 112$ which is worse than the solution we had at hand.*

3.5 Algorithm for a B-local star minimum

In this section, we will state the steps of the algorithm that are based on the extreme point approach that was presented earlier in this chapter. The steps of the algorithm goes as follows:

ALGORITHM

Initialization

Given an extreme point x^0 and one of its related bases B. Let Z^0 be the objective function value at x^0. set $k = 0$. Go to step 1.

Step 1:

Compute the reduced gradient at x^k for the non-basic variables. Sort the non-basic variables in ascending order $y_1, y_2, ..., y_{n^2-2n+1}$ with respect to their reduced gradient. Set $j = 1$. Go to step 2.

Step 2:

If $j > n^2 - 2n + 1$ (number of non-basic variables) then stop. x^k is a B-local or a relative local star minimum. Otherwise go to step 3.

Step 3:

Compute a the coefficient of δ^2 for the non basic variable y_j. Compute the ratio $-\frac{b}{a}$ where b is the reduced gradient with respect to y_j. Go to step 4.

Step 4:

If $a > 0$ and $-\frac{b}{a} > 1$, or $a < 0$ and $- < 1$, then pivot by entering the non-basic variable y_j, compute $Z^{k+1} = Z^k + a - b$, set x^{k+1} to be the new extreme point obtained after pivoting, set $k = k + 1$. Go to step 1. Otherwise set $j = j + 1$ and go to step 2.

Theorem 6 *The above algorithm converges finitely to a relative local star minimum.*

Proof. The proof is straightforward due to the fact that the objective function improves at each new extreme point visited, and since the number of extreme points is finite, we will eventually stop at a relative local star minimum. ■

Remark 5 *It is clear that, in order to be able to check whether an extreme point is a local star minimum, we have to examine all the basic representations of that extreme point. This means checking* $(2^{n-1})(n^{n-2})$ *feasible basis. This is an extremely large number even for moderate size problems (see table 2.1). This number can be reduced if we check only non-degenerate bases, the so-called alternating bases.*

3.6 Use of the characterization

The characterization obtained here may not be very efficient in solving the $\mathcal{QAP}$ optimally. This is due to the huge number of bases per extreme points, even after considering only the non-degenerate bases. Nevertheless, it remains an important result in the sense where it represents a step toward a stronger result for characterizing local star minima for the $\mathcal{QAP}$. On the other hand, the characterization may be used in designing heuristics inspired from the improvement approaches discussed in chapter 2. Example of such uses is the pair exchange strategy where in this case there is no need to explore all possible pair exchanges but rather to exchange only those pairs that would lead to a better solution.

3.7 Conclusion

In this chapter, we explored the network structure of the feasible set of the $\mathcal{QAP}$, and used a simplex like pivot to give a characterization of what we defined as relative local star minimum, that is an extreme point satisfying the local star minimum condition with respect to a given basis B. We then presented an algorithm that uses the idea developed above to obtain a relative local star minimum. This algorithm can be used as a subroutine in a more general algorithm that explores all or part of the basic representations of an extreme point to converge to an absolute local star minimum.

Chapter 4

A reformulation of the $\mathcal{QAP}$

4.1 Introduction

The quadratic assignment problem is considered as one of the hardest problems in operations research. This is due to the combinatorial nature of its feasible region. As it was stated in chapter 1, solving the $\mathcal{QAP}$ is $\mathcal{NP}$-hard and even finding an $\varepsilon-$optimal solution is $\mathcal{NP}$- hard. This difficulty resides in the fact that the solution space is highly degenerate in addition to the objective function being non-linear. Many researchers have tried to obtain a better formulation to the problem in order to ease the way to deal with it. As seen previously, a reformulation that has attracted many researchers, is the linearization of the objective function. This resulted in many formulations usually with a very large number of constraints.

In this chapter, we propose to reformulate the quadratic assignment problem with

the hope that the new formulation, will result in a small number of constraints. The new formulation is expected to provide some insights into solving and characterizing this difficult problem. The chapter is organized as follows. The motivation behind the reformulation is first stated. Next, the proposed reformulation is proposed followed by an example. Properties of the new formulation are then discussed and its equivalence with the $\mathcal{QAP}$ is proved. The chapter is concluded by a discussion of the use of the proposed formulation in Meta-heuristic schemes.

4.2 Motivation

The work in this chapter was first motivated by observing the quadratic objective function to be minimized which has the following form:

$$Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}x_{ij}x_{kl}$$

It can be observed that for a given solution, the cost $f_{ik}d_{jl}$ appears in the computation of the value of the objective function, for that given solution, only when both x_{ij}, x_{kl} are simultaneously nonzero. Then, the $\mathcal{QAP}$ will be solved optimally if we can select n arcs, n being the number of facilities to be assigned, such that:

(i) The constraints of the problem are satisfied.

(ii) The sum of the costs $f_{ik}d_{jl}$ that would appear in the computation of the value of the objective function is minimum.

Based on (i) and (ii) above, the following reformulation is made.

4.3 Reformulation of the $\mathcal{QAP}$

Using the above observations, we provide in this section a way to construct a new problem that we show is equivalent to the $\mathcal{QAP}$. In what follows, the notations $f_{i.}$ and $f_{.i}$ represent the flow from facility i to an unspecified facility represented by a dot, and respectively to facility i from an unspecified facility again represented by a dot. Similarly, the notation $d_{j.}$ and $d_{.j}$ represent the distance from location j to an unspecified location represented by a dot, and respectively the distance from an unspecified location represented by a dot to location j. Thus, $f_{i.}d_{j.}$ and $f_{.i}d_{.j}$ corresponds to the cross product of the flow from (respectively to) facility i and the distance from (respectively to) its location j with an unspecified facility and its unspecified location that are represented by dots.

Clearly, if arc (i, j) is not active in the optimal solution (its corresponding x_{ij} is zero), then the corresponding $f_{i.}d_{j.}$ and $f_{.i}d_{.j}$ will not contribute to the computation of the objective function value, this is because these coefficients are multiplied by x_{ij} which is zero. On the other hand if arc (i, j) is active in the optimal solution (its corresponding x_{ij} is one), then the corresponding $f_{i.}d_{j.}$ and $f_{.i}d_{.j}$ will contribute to the computation of the objective function value only if the dots are replaced by k and l respectively for all arcs (k, l) that are also active in the solution.(i.e. if $x_{kl} = 1$ then $f_{ik}d_{jl}$ will contribute).

This observation lead us to think that, if , in a new bipartite graph, arcs (i, j) and (k, l) are replaced by some nodes say p and q, the new nodes (that correspond to arcs in the original problem) are linked by an arc if they correspond to arcs that potentially may appear in the same assignment. The cost of such new arcs is denoted $c_{[pq]}$ and is taken such that $c_{[pq]} = f_{ik}d_{jl}$. An optimal solution would correspond to choosing n nodes, in the new problem, such that the sum of the costs of the arcs joining any two of these n nodes is minimized. So, a graph is obtained by replacing all arcs (i, j) by new nodes and duplicating the nodes, then joining these new nodes as it will be presented next.

To obtain this new graph, the steps below are to be followed:

1 Enumerate all arcs (i, j) in the original graph in an arbitrary order.

2 Create a bipartite graph with two sets $\mathcal{S}$ and $\mathcal{D}$ representing the source and destination node sets respectively. For each arc l in the original graph, create two nodes s_l and d_l in $\mathcal{S}$ and $\mathcal{D}$ respectively.

3 Any two arcs that can appear in the same feasible assignment of the original problem should have their corresponding nodes connected in the resulting graph. This step could be stated as follows: Any two nodes s_i and d_j, in $\mathcal{S}$ and $\mathcal{D}$ respectively, in the resulting graph should not be connected if in the original graph their corresponding arcs i and j are incident to the same node. Other nodes should be connected.

In other words, for each node $i \in S$, let C_i be the set of all nodes whose corresponding arcs in the original problem that are incident to the extremities of the arc, in the original graph, corresponding to node i. Nodes in C_i should not be connected, while the remaining nodes should be.

The following example shows how the bipartite graph is obtained.

4.4 Example

In what follows we give a graphical example of the $\mathcal{QAP}$ and its reformulation using the steps described above.

Given a $\mathcal{QAP}$

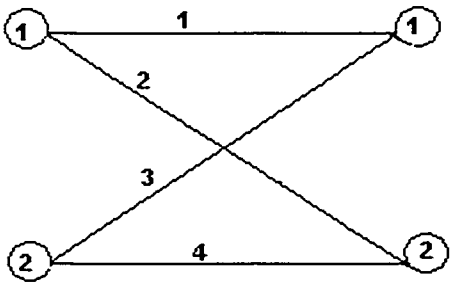

Figure 4.1: A 2×2 assignment

First we enumerate the arcs as specified in the first step of the reformulation process (The numbers are represented on the arcs in figure 4.1).

Applying the second and third steps of the reformulation process, we obtain the following figure:

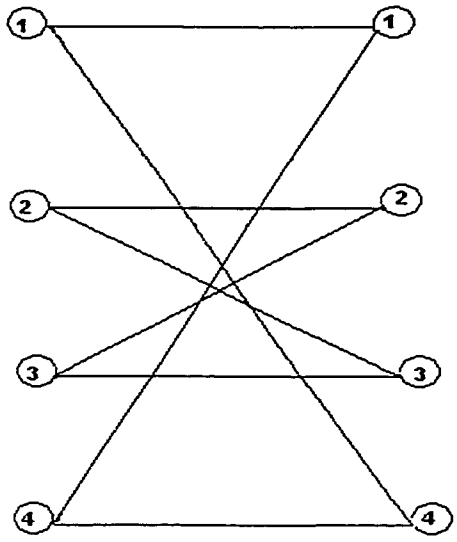

Figure 4.2: The new graph

A mathematical formulation for the resulting network is given below.

P 4.1

$$\textit{minimize} \qquad Z(x) = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{[ij]} x_{[ij]}$$

$$\textit{subject to} \ :$$

$$\sum_{i \in \mathcal{S}} x_{[ij]} = (n) y_j \qquad \forall j \in \mathcal{D}$$

$$\sum_{j \in \mathcal{D}} x_{[ij]} = (n) y_i \qquad \forall i \in \mathcal{S}$$

$$\sum_{i \in C_i} y_i = 1 \qquad \forall i \in S$$

$$x_{[ij]} \text{ and } y_i \in \{0, 1\}, \forall i \in \mathcal{S}, j \in \mathcal{D}$$

where

$$x_{[ij[} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is selected in the solution} \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_i = \begin{cases} 1 & \text{if node } i \text{ is selected in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$x_{[ij]}$ corresponds to an arc adjacent to nodes i and j in (**P 4.1**), where in the original $\mathcal{QAP}$ it corresponds to the two arcs that were numbered i and j respectively.

The costs $c_{[ij]}$ of arc (i,j) in the new formulation (**P 4.1**), corresponds to $f_{pk}d_{ql}$ in the original $\mathcal{QAP}$, here i in (**P 4.1**) corresponds to arc (p,q) in the original $\mathcal{QAP}$ and in a same way j corresponds to arc (k,l) in the original $\mathcal{QAP}$.

We can see that there is a one to one relation between y_i and $x_{[ii]}$, that is, if y_i is 1 (0) this means that the corresponding arc i in the original problem is active (non active). Hence, node i in the new problem is chosen (not chosen). This means that the arc (i,i) in the new problem is active (non active). Hence $x_{[ii]}$ is 1 (0). Taking this into consideration, the formulation can slightly be modified to obtain:

P 4.2

$$\textit{minimize} \qquad Z(x) = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{[ij]}x_{[ij]}$$

$$\textit{subject to} \ :$$

$$\sum_{\substack{i\in\mathcal{S}\\ i\neq j}} x_{[ij]} = (n-1)y_j \qquad \forall j \in \mathcal{D}$$

$$\sum_{\substack{j\in\mathcal{D}\\ i\neq j}} x_{[ij]} = (n-1)y_i \qquad \forall i \in \mathcal{S}$$

$$\sum_{i\in C_i} y_i = 1 \qquad \forall i \in S$$

$$x_{[ij]} \text{ and } y_i \in \{0,1\} \qquad \forall i \in \mathcal{S}, j \in \mathcal{D}$$

We can notice that if the values of the y_i's are fixed, the resulting is a pure network

problem. We call such a problem a *"variable network"*.

4.5 A numerical example

In what follows we give a numerical example of the reformulation of a $\mathcal{QAP}$.

Given a $\mathcal{QAP}$ with flow and distance matrices F and D respectively given below:

$$F = \begin{vmatrix} 0 & 4 & 3 \\ 4 & 0 & 3 \\ 3 & 3 & 0 \end{vmatrix} \qquad D = \begin{vmatrix} 0 & 1 & 3 \\ 1 & 0 & 5 \\ 3 & 5 & 0 \end{vmatrix}$$

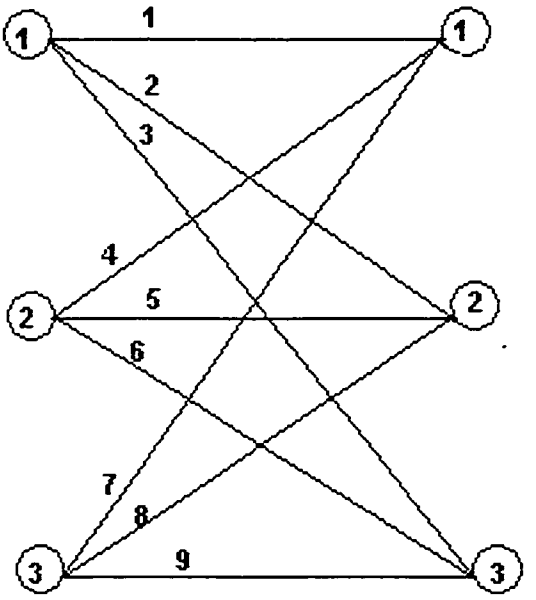

Figure 4.3: Graphical representation of the example

The numbers on the arcs correspond to the arc numbers (first step of the reformulation process). Using the steps of the reformulation process above, we obtain the following graph for the reformulated problem:

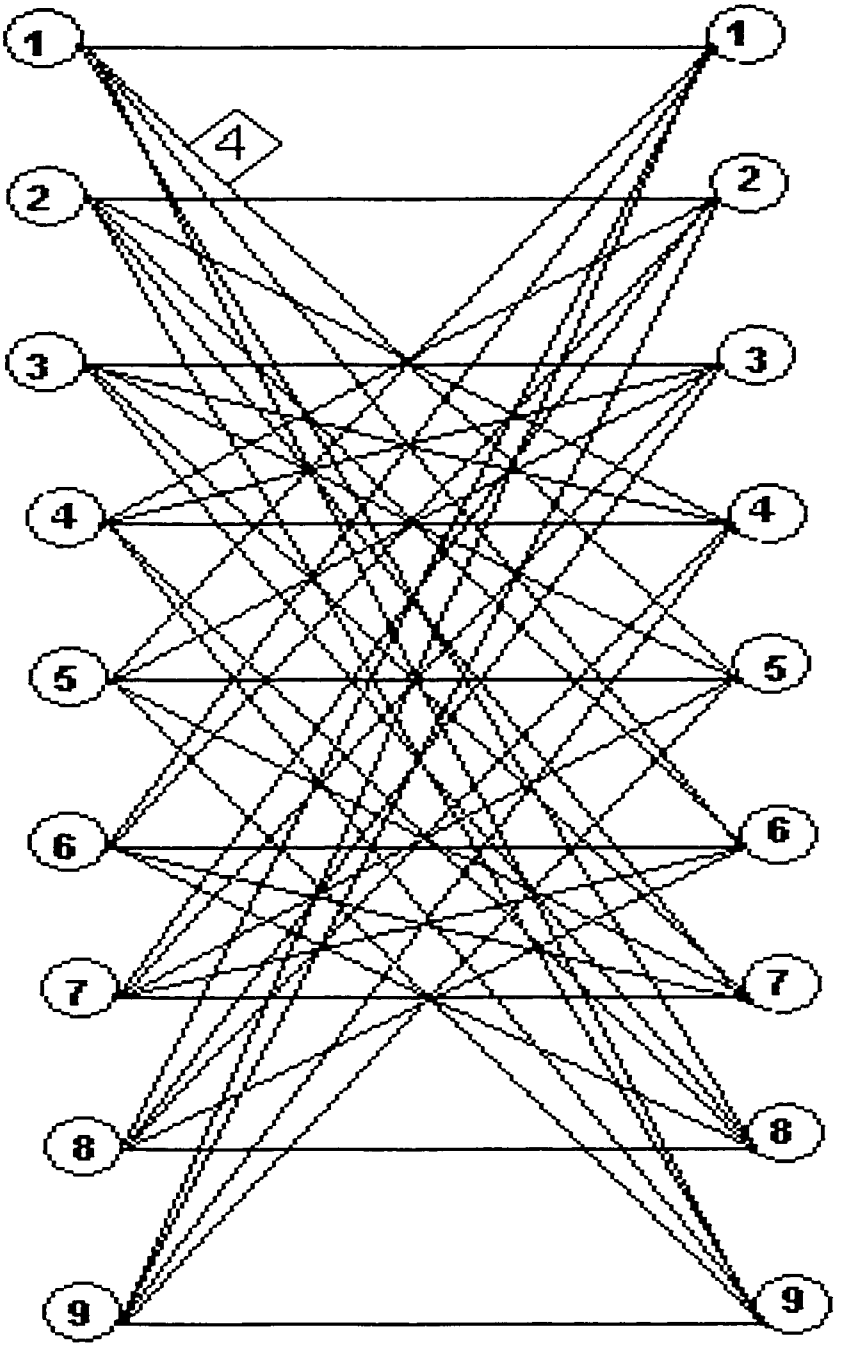

Figure 4.4: Graph of the reformulated problem

The cost shown on arc $(1,5)$ in the graph is $c_{[15]} = f_{12}d_{12} = 4$, where nodes 1 and 5 correspond to arcs $(1,1)$ and $(2,2)$ respectively in the original problem. The remaining costs are given in the following cost matrix:

$$C = \begin{vmatrix} 0 & \infty & \infty & \infty & 4 & 12 & \infty & 3 & 9 \\ \infty & 0 & \infty & 4 & \infty & 20 & 3 & \infty & 15 \\ \infty & \infty & 0 & 12 & 20 & \infty & 9 & 3 & \infty \\ \infty & 4 & 12 & 0 & \infty & \infty & \infty & 3 & 9 \\ 4 & \infty & 20 & \infty & 0 & \infty & 3 & \infty & 15 \\ 12 & 20 & \infty & \infty & \infty & 0 & 9 & 15 & \infty \\ \infty & 3 & 9 & \infty & 3 & 9 & 0 & \infty & \infty \\ 3 & \infty & 3 & 3 & \infty & 15 & \infty & 0 & \infty \\ 9 & 15 & \infty & 9 & 15 & \infty & \infty & \infty & 0 \end{vmatrix}$$

Arcs that do not appear in the graph of the new problem have been assigned an infinite cost.

Remark 6 *In the above reformulation, we only considered $\mathcal{QAP}$'s where the objective function is purely quadratic (no linear term). In case where a linear term is present, the cost $c_{[ii]}$ (that are zero in our example) on the arc (i,i) in the new problem would correspond to the linear cost t_{kl} in the original problem (Here, arc (k,l) in the original problem is denoted i in the reformulated problem).*

Hence we get the following optimization problem:

minimize

$$\begin{aligned}
Z(x) = {} & 4x_{[15]} + 12x_{[16]} + 3x_{[18]} + 9x_{[19]} + 4x_{[24]} + 20x_{[26]} + 3x_{[27]} + 15x_{[29]} \\
& +12x_{[34]} + 20x_{[35]} + 9x_{[37]} + 3x_{[38]} + 4x_{[42]} + 12x_{[43]} + 3x_{[48]} + 9x_{[49]} \\
& +4x_{[51]} + 20x_{[53]} + 3x_{[57]} + 15x_{[59]} + 12x_{[61]} + 20x_{[62]} + 9x_{[67]} + 15x_{[68]} \\
& +3x_{[72]} + 9x_{[73]} + 3x_{[75]} + 9x_{[76]} + 3x_{[81]} + 3x_{[83]} + 3x_{[84]} + 15x_{[86]} \\
& +9x_{[91]} + 15x_{[92]} + 9x_{[94]} + 15x_{[95]} \\
& +\mathrm{M}[x_{[12]} + x_{[13]} + x_{[14]} + x_{[17]} + x_{[21]} + x_{[23]} + x_{[25]} + x_{[28]} + x_{[31]} + x_{[32]} \\
& +x_{[36]} + x_{[39]} + x_{[41]} + x_{[45]} + x_{[46]} + x_{[47]} + x_{[52]} + x_{[54]} + x_{[56]} + x_{[58]} \\
& +x_{[63]} + x_{[64]} + x_{[65]} + x_{[69]} + x_{[71]} + x_{[74]} + x_{[78]} + x_{[79]} + x_{[82]} + x_{[85]} \\
& +x_{[87]} + x_{[89]} + x_{[93]} + x_{[96]} + x_{[97]} + x_{[98]}]
\end{aligned}$$

subject to :

$$\begin{aligned}
& x_{[12]} + x_{[13]} + x_{[14]} + x_{[15]} + x_{[16]} + x_{[17]} + x_{[18]} + x_{[19]} = 2y_1 \\
& x_{[21]} + x_{[23]} + x_{[24]} + x_{[25]} + x_{[26]} + x_{[27]} + x_{[28]} + x_{[29]} = 2y_2 \\
& x_{[31]} + x_{[32]} + x_{[34]} + x_{[35]} + x_{[36]} + x_{[37]} + x_{[38]} + x_{[39]} = 2y_3 \\
& x_{[41]} + x_{[42]} + x_{[43]} + x_{[45]} + x_{[46]} + x_{[47]} + x_{[48]} + x_{[49]} = 2y_4 \\
& x_{[51]} + x_{[52]} + x_{[53]} + x_{[54]} + x_{[56]} + x_{[57]} + x_{[58]} + x_{[59]} = 2y_5 \\
& x_{[61]} + x_{[62]} + x_{[63]} + x_{[64]} + x_{[65]} + x_{[67]} + x_{[68]} + x_{[69]} = 2y_6 \\
& x_{[71]} + x_{[72]} + x_{[73]} + x_{[74]} + x_{[75]} + x_{[76]} + x_{[78]} + x_{[79]} = 2y_7 \\
& x_{[81]} + x_{[82]} + x_{[83]} + x_{[84]} + x_{[85]} + x_{[86]} + x_{[87]} + x_{[89]} = 2y_8 \\
& x_{[91]} + x_{[92]} + x_{[93]} + x_{[94]} + x_{[95]} + x_{[96]} + x_{[97]} + x_{[98]} = 2y_9 \\
& x_{[21]} + x_{[31]} + x_{[41]} + x_{[51]} + x_{[61]} + x_{[71]} + x_{[81]} + x_{[91]} = 2y_1 \\
& x_{[12]} + x_{[32]} + x_{[42]} + x_{[52]} + x_{[62]} + x_{[72]} + x_{[82]} + x_{[92]} = 2y_2 \\
& x_{[13]} + x_{[23]} + x_{[43]} + x_{[53]} + x_{[63]} + x_{[73]} + x_{[83]} + x_{[93]} = 2y_3 \\
& x_{[14]} + x_{[24]} + x_{[34]} + x_{[54]} + x_{[64]} + x_{[74]} + x_{[84]} + x_{[94]} = 2y_4 \\
& x_{[15]} + x_{[25]} + x_{[35]} + x_{[45]} + x_{[65]} + x_{[75]} + x_{[85]} + x_{[95]} = 2y_5 \\
& x_{[16]} + x_{[26]} + x_{[36]} + x_{[46]} + x_{[56]} + x_{[76]} + x_{[86]} + x_{[96]} = 2y_6 \\
& x_{[17]} + x_{[27]} + x_{[37]} + x_{[47]} + x_{[57]} + x_{[67]} + x_{[87]} + x_{[97]} = 2y_7 \\
& x_{[18]} + x_{[28]} + x_{[38]} + x_{[48]} + x_{[58]} + x_{[68]} + x_{[78]} + x_{[98]} = 2y_8 \\
& x_{[19]} + x_{[29]} + x_{[39]} + x_{[49]} + x_{[59]} + x_{[69]} + x_{[79]} + x_{[89]} = 2y_9 \\
& y_1 + y_2 + y_3 = 1 \\
& y_4 + y_5 + y_6 = 1 \\
& y_7 + y_8 + y_9 = 1 \\
& y_1 + y_4 + y_7 = 1 \\
& y_2 + y_5 + y_8 = 1 \\
& y_3 + y_6 + y_9 = 1 \\
& x_{[ij]} \text{ and } y_i \in \{0, 1\} \qquad i, j = 1...9
\end{aligned}$$

The above problem was solved using **Extended LINDO 6.1** [Licence # LDPC5-611050]. We obtained the following solution

$$\begin{cases} x_{[15]} = x_{[19]} = x_{[51]} = x_{[59]} = x_{[91]} = x_{[95]} = 1 \\ x_{[ij]} = 0 \qquad\qquad\qquad\qquad \text{Otherwise} \end{cases}$$

This solution corresponds to selecting arcs $1, 5$ and 9, this corresponds to the following solution of the original $\mathcal{QAP}$:

$$\begin{cases} x_{11} = x_{22} = x_{33} = 1 \\ x_{ij} = 0 \qquad \text{Otherwise} \end{cases}$$

Which is indeed the optimal solution for the problem in the example.

4.6 Properties and equivalence

Lemma 1 *In the above formulation, integrality of y_i's implies integrality of $x_{[ij]}$'s*

Proof.

The set of constraints representing $x_{[ij]}$ corresponds to a unimodular matrix, since we have a structure of the following form:

$$\begin{aligned} \sum_{i \in \mathcal{S}} x_{[ij]} &= b_j \qquad \forall j \in \mathcal{D} \\ \sum_{j \in \mathcal{D}} x_{[ij]} &= b_i \qquad \forall i \in \mathcal{S} \end{aligned}$$

Hence, by imposing integrality on the right-hand side, this would give us an integer solution for the $x_{[ij]}$'s. ■

Hence we get the following formulation that relaxes problem

P 4.2:

$$\textit{minimize} \qquad Z(x) = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{[ij]}x_{[ij]}$$

$$\textit{subject to} \ :$$

$$\sum_{i\in\mathcal{S}} x_{[ij]} = (n-1)y_j \qquad \forall j \in \mathcal{D}$$

$$\sum_{j\in\mathcal{D}} x_{[ij]} = (n-1)y_i \qquad \forall i \in \mathcal{S}$$

$$\sum_{i\in C_i} y_i = 1 \qquad \forall i \in S$$

$$x_{[ij]} \in [0,1] \qquad \forall i \in \mathcal{S}, j \in \mathcal{D}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{S}$$

Now, letting P_1 represent the original quadratic assignment problem and, let P_2 represent the new problem obtained by applying the above transformation steps. We give the following definitions that will be useful in the continuation of the proposed formulation.

Definition 15 *Given the quadratic assignment problem P_1. A feasible assignment is a set of pairs (i,j) satisfying:*

- Every node $i \in \mathcal{S}$ has exactly one outgoing arc (i,j)

- Every node $j \in \mathcal{D}$ has exactly one incoming arc (i,j)

Definition 16 *Given the transformed problem P_2. A node $i \in \mathcal{S}$ (or D) is said to be covered by node $j \in \mathcal{S}$ (or D) if there is a side edge (i,j) or (j,i) connecting*

them, where a side edge is an edge linking nodes in $\mathcal{S}$ or nodes in $\mathcal{D}$.

Definition 17 *Given the transformed problem P_2. A feasible solution is a set of nodes $I_S \cup I_D$ such that $I_S \subset S$, $I_D \subset D$, and $\| I_S \| = \| I_S \| = n$, where n is the size of the original $\mathcal{QAP}$, satisfying the following:*

- Every node $i \in I_S$ is connected to all nodes in I_D.

- Every node $j \in I_D$ is connected to all nodes in I_S.

- Every node $i \in S - I_S$ is covered by a node in I_S.

- Every node $j \in D - I_D$ is covered by a node in I_D.

Remark 7 *If we connect any two nodes s_i and s_j in $\mathcal{S}$, in the resulting graph, if they are respectively connected to d_j and d_i graph representing a feasible solution to the $\mathcal{QAP}$ in the new formulation would correspond to a clique. This is illustrated in figure 4.5 that corresponds to the graph of the first example after adding side arcs.*

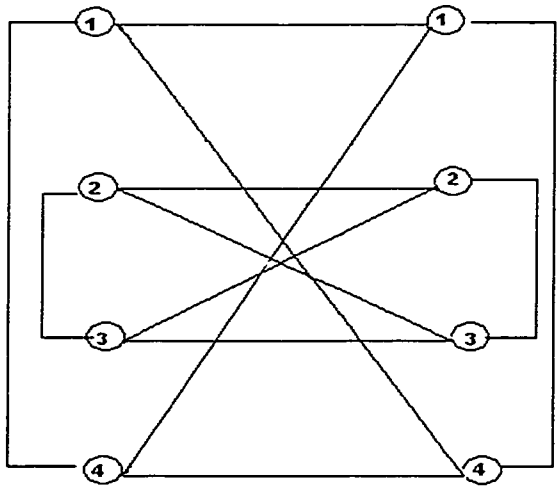

Figure 4.5: The graph after adding side arcs

Definition 18 *Given a graph $G = (X, A)$, a clique is a subgraph $G_c = (X_c \subset X, A_G \subset A)$ of G, where an arc exist between every of nodes of X_g.*

In what follows, we prove the equivalence of P_1 and P_2.

Theorem 7 *The two problems P_1 and P_2 stated above are equivalent.*

Proof.

The proof of the theorem follows directly from the following two propositions. ■

Proposition 2 *To any feasible solution in P_1 there corresponds a feasible solution in P_2.*

Proof. Feasibility in $P_1 \Rightarrow$ Feasibility in P_2?

Let $\mathcal{A}$ be a feasible assignment in P_1, then $\mathcal{A}$ contains n arcs (that do not enter or leave the same node, this is because of the feasibility of the assignment). Hence, for each arc in $\mathcal{A}$ we can choose its corresponding nodes in P_2 (one in S and one in D), this gives us a set of $2n$ nodes, n of them in S and their corresponding ones in D, Which corresponds to two subsets S_0 and D_0 respectively. Each node in S_0 is connected to all the nodes in D_0. Also, each node in S_0 (respectively in D_0) is side connected to all other nodes in S_0 (respectively in D_0) since they correspond to arcs representing a feasible assignment. Hence we get a feasible solution in P_2.

Feasibility in $P_2 \Rightarrow$ Feasibility in P_1?

Let $\mathcal{A}$ be a feasible solution in P_2, we have exactly n nodes in S (and their corresponding ones in D). All of these nodes are side connected [by definition of

feasibility in P_2]. Hence choosing any pair these nodes, their corresponding arcs in P_1 will not have a common node, and since we have n of such arcs, they will correspond to a feasible assignment in P_1. ■

Note: It is clear that the cost of an assignment in P_1 is equal to the cost of a solution in P_2.

Proposition 3 *To an optimal solution in P_1 there corresponds an optimal solution in P_2.*

Proof. In what follows, a pair (n_S, n_D) of nodes in P_2 represents a node n_S in S and its corresponding node n_D in D.

Optimality in $P_2 \Rightarrow$ Optimality in P_1?

Let $\mathcal{A}_1$ be an optimal solution in P_1, and suppose that $\mathcal{A}_2$ its corresponding solution in P_2 is not optimal. Hence there exists two pairs of nodes (n_S^1, n_D^1) and (n_S^2, n_D^2) that can be exchanged with two other pairs of nodes (v_S^1, v_D^1) and (v_S^2, v_D^2) (exchanging only one pair of nodes would violate feasibility of the solution) such that the cost of the new solution after the exchange in less that the cost of the solution before the exchange (i.e. $\mathcal{A}_2$). Hence the cost of the assignment in P_1corresponding to the solution in P_2 after the exchange will have lower cost that the cost of $\mathcal{A}_1$, which contradicts the assumption of optimality of $\mathcal{A}_1$.

Optimality in $P_1 \Rightarrow$ Optimality in P_2?

Let $\mathcal{A}_2$ be an optimal solution in P_2, and suppose that $\mathcal{A}_1$ its corresponding

assignment in P_1 is not optimal. Hence there exists two pairs of arcs (i_1, j_1) and (i_2, j_2) that can be exchanged with two other pairs of arcs (i'_1, j'_1) and (i'_2, j'_2) (exchanging only two arcs would violate feasibility of the assignment) such that the cost of the new assignment after the exchange of arcs in less that the cost of the assignment before the exchange (i.e. $\mathcal{A}_1$). Hence the cost of the solution in P_2 corresponding to the assignment in P_1 after the exchange will have lower cost that the cost of $\mathcal{A}_2$, which contradicts the assumption of optimality of $\mathcal{A}_2$. ■

4.7 Characteristics

As it was mentioned before, the new formulation corresponds to an aggregation of Adams and Johnson's $\mathcal{RLT}$ based formulation given in **P 2.10**. In terms of variables and constraints we have the following:

There is n^4 continuous variables, in the interval $[0, 1]$, of the form $x_{[ij]}$ and n^2 integer variables in $\{0, 1\}$ of the form y_i. Hence the total number of variables is:

$$\text{Total number of variables} = n^4 + n^2 = n^2(n^2 + 1)$$

Concerning the number of constraints we have:

n^2 constraints of the form $\sum_{i \in \mathcal{S}} x_{[ij]} = (n-1)y_j$ and n^2 constraints of the form $\sum_{j \in \mathcal{D}} x_{[ij]} = (n-1)y_i$. Hence $2n^2$ constraints. On the other hand we have $2n$ constraints of the form $\sum_{i \in C_i} y_i = 1$. This in addition to constraints of the types $x_{[ij]} \in [0, 1]$ or $y_i \in \{0, 1\}$. Hence without counting the bounds and integrality con-

straints on the variables, the total number of constraints is given by:

$$\text{Total number of constraints} = 2n^2 + 2n = 2n(n+1)$$

4.8 Discussion

The above formulation has been used for some test problems. It turned out that the linear relaxation gave a weak lower bound compared to bounds obtained by other formulations such as the one obtained by Adams and Johnson's (**P 2.10**). In fact, the above formulation corresponds to an aggregation of Adams and Johnson's formulation (**P 2.10**) obtained using the Relaxation Linearization Technique ($\mathcal{RLT}$) of Sherali and Adams [123] that we discuss later. This explains the weakness of the bound obtained compared at least to the bound obtained using the $\mathcal{RLT}$ relaxation. Although the bound obtained is weak, the formulation still has an undeniable advantage, since by fixing the y_i's, we remain with a network structure that is easily dealt with. Hence this formulation could be used in a heuristic approach to solve the problem. In the following we suggest two approaches where the above formulation can be used.

Tabu search.

A tabu search heuristic can be designed using the suggested formulation. In this heuristic one can fix $n-k$,where $2 \leq k < n$, y_i's then solve for the rest of the

variables to an integral optimality. Generating neighbors would be done by feasibly interchanging two or more of the $n-k$ fixed variables and again solve to optimality.

Genetic Algorithm.

Another heuristic using Genetic Algorithms can also be designed using the suggested formulation. In this case an initial generation can be obtained by randomly fixing $n-k$,where $2 \leq k < n$, y_i's then solving to optimality for each case, the next generation can be obtained from the previous one by forcing the variables with higher reduced cost into the solution.

4.9 Conclusion

In this chapter, the $\mathcal{QAP}$ is reformulated as a constrained network problem, we called it **Variable Network**. The properties of the proposed formulation are investigated and eventual uses of the proposed formulation in designing heuristic algorithms for the $\mathcal{QAP}$ are also discussed. It may be noted that the proposed formulation gives a lower bound for the $\mathcal{QAP}$. Although this bound is weaker that known bounds, it may still be useful.. The usefulness of this bound is discussed in the last chapter.

Chapter 5

General forms of the $\mathcal{QAP}$

5.1 Introduction

The $\mathcal{QAP}$ as discussed in the previous chapters has received considerable attention in the last four decades. Numerous approaches have been developed for solving this problem. Although, the $\mathcal{QAP}$ as formulated has many applications, it does not represent the most general form of facility location problems. This formulation has many limitations that restrict its applications. These limitations include:

- Only one facility can be assigned to a location. This limits the opportunity of locating several facilities at a single location. Although this limitation may be overcome with the actual formulation, by duplicating such locations as many times as needed, this would increase the size of the problem and limit the possibility of solving large problems. The largest size of $\mathcal{QAP}$ that can be

solved in a reasonable amount of time is of size $18 - 20$.

- The number of facilities to be assigned to each location is fixed. This number is not determined through the optimization process. In many cases, locations may accommodate more facilities than specified in $\mathcal{QAP}$.

Allowing this to happen may complicate the problem, but it is expected to enrich the applications and identify new opportunities for optimization.

In this chapter, we generalize the $\mathcal{QAP}$ formulation to remedy the two points above , and give general models for assigning facilities to locations. We then show that the $\mathcal{QAP}$ and other related problems are special cases of the proposed formulation. The chapter is organized as follows: in section 5.2 the generalization of the $\mathcal{QAP}$ is presented. In the next section some special cases of the generalization are discussed followed by some related problems in section 5.4. The complexity of the proposed problem is discussed in section 5.5. Finally a brief conclusion is given in the last section.

5.2 Generalizations of the $\mathcal{QAP}$

Consider the problem of assigning n facilities $i = 1...n$ to m locations $j = 1, ..., m$. Each location j can accommodate up to n_j facilities, and the total capacity of all locations is such that:

$$\sum_{j=1}^{m} n_j \geq n$$

That is, all facilities can be located at some location. A flow between any two facilities i and k is given and denoted f_{ik}. The distance between any two locations j and l is known and denoted d_{jl}. Also, a cost c_{ij} is incurred if facility i is located at location j, this cost could represent the construction cost or any other cost that depends only on the type of facility and the location where it is assigned.

The objective is to find the optimal assignment of all facilities to the possible locations to minimize a quadratic objective function. The quadratic term corresponding to the product of the inter-flow between two facilities and the distance between the locations where the facilities are to be located. This term is augmented by the flow between facilities located at the same location. A linear term is added, this term represents the cost of assigning a facility to a location.

Let us define the variables x_{ij}, such that:

$$x_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is assigned to location } j \\ 0 & \text{Otherwise} \end{cases} \qquad i = 1...n,\ j = 1, ..., m$$

A formal representation of the above problem is referred to as [P 1] and is given as:

P 5.1

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

subject to :

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1...n \tag{5.1}$$

$$\sum_{i=1}^{n} x_{ij} \leq n_j \qquad \forall j = 1, ..., m \tag{5.2}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1...n,\ j = 1, ..., m$$

Constraint (5.1) represents the condition that all facilities must be located, constraint (5.2) ensures that no location receives more than its capacity. The inequality in (5.2) can be transformed into an equality by introducing slack variables, in that case we get the following problem.

P 5.2

$$\textit{minimize} \quad \sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

subject to :

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1...n \tag{5.3}$$

$$\sum_{i=1}^{n} x_{ij} + s_j = n_j \qquad \forall j = 1, ..., m \tag{5.4}$$

$$x_{ij} \in \{0, 1\}, s_j \geq 0 \text{ and integer}$$

s_j represents the unused capacity of each location. The above problem can be represented differently by introducing a variable z_j, instead of s_j that represents the unused capacity at location j, that would represent the number of facilities that are to be assigned to the location j, if we need to decide on this number through the optimization process. In this case we get the following formulation:

P 5.3

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

subject to :

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1...n \tag{5.5}$$

$$\sum_{i=1}^{n} x_{ij} - z_j = 0 \quad \forall j = 1, ..., m \tag{5.6}$$

$$z_j \leq n_j \qquad \forall j = 1, ..., \tag{5.7}$$

$$x_{ij} \in \{0, 1\}, z_j \geq 0 \text{ and integer}$$

Remark 8 *In case where no limitations are imposed on the capacity of the locations, the last sets of constraints 5.2 and 5.4 would be dropped from* ***(P 5.1)*** *and* ***(P 5.2)*** *respectively. In problem* ***(P 5.3)*** *the constraints 5.7 representing the upper bounds on the number of facilities to be assigned to the same location would be dropped.*

In the three above cases, we can think of adding a cost proportional to the number of facilities assigned to the same location, say t_j. This may be thought of as a penalty to force fairness. I.e. we do not want all facilities in the same location. In this case a fourth term is added to the objective function. Hence the objective function in (**P 5.1**) and (**P 5.2**) would be:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{j=1}^{m} t_j \sum_{i=1}^{n} x_{ij}$$

In the third case [i.e. problem (**P 5.3**)], we would get:

$$\min \; z = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{j=1}^{m} t_j z_j$$

The set of constraints in all three cases would remain unchanged.

Problem (**P 5.1**) and its derivatives given in (**P 5.2**) and (**P 5.3**) will be referred to as the "*Generalized Quadratic Assignment Problem*" or $\mathcal{GQAP}$. A special case, referred to as the " *Generalized Quadratic Semi-Assignment Problem*" or $\mathcal{GQSAP}$ is given in the next subsection.

5.3 Special cases of $\mathcal{GQAP}$

In this section an account of several special cases of $\mathcal{GQAP}$ are proposed.

5.3.1 The Generalized Quadratic Semi-Assignment Problem.

In the case where n_j, the number of facilities to be assigned to each location j, is known and that the total capacity of the locations is equal the number of facilities i.e.

$$\sum_{j=1}^{m} n_j = n$$

the problem (**P 5.1**) would reduce to problem (**P 5.4**) given below:

P 5.4

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

subject to :

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1...n \tag{5.8}$$

$$\sum_{i=1}^{n} x_{ij} = n_j \qquad \forall j = 1, ..., m \tag{5.9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1...n,\ j = 1, ..., m$$

The feasible set in this case corresponds to the semi-assignment polytope. For this reason, we referred to the problem (**P 5.4**) as the " *Generalized Quadratic Semi-Assignment Problem*" or $\mathcal{GQSAP}$.

5.3.2 The Quadratic Assignment Problem

As it was mentioned before, the $\mathcal{QAP}$ is a special case of the problems introduced above. We can easily see that the $\mathcal{QAP}$ is a special case of $\mathcal{GQSAP}$. It is obtained from (**P 5.4**) by setting $m = n$ and letting $n_j = 1$ for each facility j. Since, when $n_j = 1$, no two facilities can be located at the same location this would make the product $x_{ik}x_{jk} = 0$, hence dropping the second part of the quadratic term, leaving us only with the objective function of the classic $\mathcal{QAP}$. Also, $\mathcal{GQSAP}$ can be transformed into a $\mathcal{QAP}$ by duplicating each facility j, n_j times.

5.3.3 The Quadratic Semi-Assignment problem

The Quadratic semi-assignment problem has been formulated as follows:

$$QAP \begin{cases} \textit{minimize } \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} f_{ij}x_{ik}x_{jk} \\ \text{subject to :} \\ \qquad \sum_{j=1}^{m} x_{ij} = 1 \quad \forall i = 1...n \\ \qquad x_{ij} \in \{0,1\} \ \forall i = 1...n, \ j = 1,...,m \end{cases}$$

This formulation is obtained from the formulation in (**P 5.1**) if we make the following assumptions:

- There is no limit on the number of facilities to be assigned to a given location (i.e. n_j is assumed to be infinite).
- There is no flow between facilities located at different locations (i.e. $f_{ik}x_{ij}x_{kl} = 0$ if $j \neq l$).

- No assignment cost is assumed (i.e. $c_{ij} = 0$).

Special cases of the quadratic semi-assignment problem include the clustering problem, the equipartition problem and the m-coloring problem on graphs. Since these problems are special cases of the quadratic semi-assignment problem, and since the quadratic semi-assignment problem is a special case of the generalized quadratic semi-assignment problem, they are then special cases of the generalized quadratic semi-assignment problem.

The clustering problem:

In the clustering problem, we are given n patterns and an $n \times n$ dissimilarity matrix $D = (d_{ij})$, the objective is to group the patterns into m clusters (groups) that minimizes the dissimilarities between patterns in the same cluster.

The equipartition problem:

Given n objects with weights w_i, $i = 1...n$, the equipartition problem is to find m classes so as to minimize the variance of the class weights.

The m-coloring problem:

Given a graph $G(V, A)$, the graph admits a coloration of its vertices with m colors, where no two adjacent vertices have the same color, if and only if the following problem:

$$m\text{-colorin}\begin{cases}\min \sum_{k=1}^{m}\sum_{(i,j)\in A} x_{ik}x_{jk} \\ \sum_{k=1}^{m} x_{jk}=1 \quad \forall j=1...n \\ x_{ik}\in\{0,1\}\end{cases}$$

has optimal function value zero.

5.4 Related problems to $\mathcal{GQAP}$

5.4.1 The quadratic set covering problem

The $\mathcal{QSCP}$ can be defined as follows: Let X be an n vector, Q be an $n \times n$ matrix, and A an $m \times n$ matrix of 0 and $1's$, the $\mathcal{QSCP}$ is formulated as follows:

$$QSCP\begin{cases}\min X^tQX \\ AX \geq e_m \\ X \text{ binary}\end{cases}$$

where e_m represents a vector of m ones.

The problem attempts to find the minimum cost of covering m markets from n warehouses. The markets may be served by more than one warehouse, as long as it is covered by at least one. The entries of the matrix A are binary such that :

$$a_{ij}=\begin{cases}1 \text{ if warehouse } i \text{ can cover market } j \\ 0 \text{ Otherwise}\end{cases}$$

5.4.2 The quadratic set partitioning problem

This is a special case of $\mathcal{QSCP}$. In this case each market is required to be covered by exactly one warehouse. It is formulated as follows:

$$QSPP \begin{cases} \min X^t Q X \\ AX = e_m \\ X \text{ binary} \end{cases}$$

5.5 Complexity of the problems

As seen above, the *Generalized Quadratic Assignment Problem* represents a generalization to many well known $\mathcal{NP}$-hard problem. In particular, it was established that the $\mathcal{QAP}$, that was proven to be $\mathcal{NP}$-hard, is nothing but a special case of it. So it is expected that the $\mathcal{GQAP}$ is indeed an $\mathcal{NP}$-hard problem.

To prove that a given problem P is $\mathcal{NP}$-hard, the usual way is to show that a proven $\mathcal{NP}$-hard problem converts to problem P in polynomial time. We use this to establish the following result.

Theorem 8 *The Generalized Quadratic Assignment Problem is $\mathcal{NP}$-hard.*

Proof. Consider the following transformation of the Quadratic Assignment Problem: Add as many dummy locations as needed, with very high or infinite distance from all the locations of the problem. Then make the cost of locating facilities at these dummy locations to be very high or infinity. Clearly this would not make any changes

to the solution of the $\mathcal{QAP}$, since any location of a facility to these dummy locations is forbidden by the very high cost of assignment and the long distance that needs to be traveled. Now, we can increase the capacities of such locations to any number greater than one. The obtained problem is clearly of the form of the $\mathcal{GQAP}$ given in [P 1]. It is clear that this transformation can be done in polynomial time, since we only need to visit each location once. Hence we obtained a transformation of the $\mathcal{QAP}$ into $\mathcal{GQAP}$ in polynomial time. Since the $\mathcal{QAP}$ is $\mathcal{NP}$-hard then we conclude that $\mathcal{GQAP}$ is also $\mathcal{NP}$-hard. ■

5.6 Conclusion

In this chapter, more general problems in the area of facility location are presented. These problems represent a generalization of the well known Quadratic Assignment Problem and some of its related problems. Such a generalization opens new horizons for research in the area of combinatorial optimization. A study of the complexity of the problem showed that, as it was expected, the problem is $\mathcal{NP}$-hard.

Chapter 6

Branch and Bound Algorithm

6.1 Introduction

The purpose of this chapter is to present several branch and bound algorithms to solve the $\mathcal{GQSAP}$ introduced in the previous chapter. In this chapter we adapt and extend several approaches that are used for developing branch and bound algorithms for the $\mathcal{QAP}$. These approaches include starting solutions and lower bounds.

Branch and bound based algorithms play a major role in designing solution algorithms for the quadratic assignment problem. In fact, more than 80% of the solutions designed for the $\mathcal{QAP}$ are based on selective enumeration. Lower bounds on the other hand, have a crucial importance for branch and bound algorithms. They are important because they restrict the enumeration of solutions to only those potentially optimal. In this chapter we discuss one of the first and the most used lower bounds

for the quadratic assignment problem namely the Gilmore's bound. We adopt such a bound to the newly introduced problem, that is the generalized quadratic semi-assignment problem. This chapter is organized as follows: in section 6.2 various lower bounds are presented and their adaptation to $\mathcal{GQSAP}$ is discussed. Section 6.3 discusses two starting solution and their adaptation to our problem. A branch and bound algorithm is presented in section 6.4 together with results obtained using the discussed lower bounds and starting solutions. Finally a conclusion is presented in section 6.5.

6.2 Gilmore's Bounds

Gilmore in [56] introduced two ways of computing a lower bound for Koopmans and Beckmann problem (**P 1.2**). The first one is obtained by rearranging the elements of the flow and distance matrices. The second one is obtained by solving a single assignment problem, for which the costs are obtained again by rearranging the elements of the later matrices. This second way of computing the lower bound provided a better lower bound than the first one. Independently from Gilmore, Lawler[86] proposed a lower bound for the Lawler's formulation (**P 2.4**), this later turned out to be equivalent to the second Gilmore's bound in the case of the Koopmans and Beckmann problem (**P 1.2**). This bound is obtained by solving $n^2 + 1$ successive linear assignment problems. The first n^2 linear assignments compute the costs for each of the n^2 variables, the last one uses these costs to obtain a lower bound.

6.2.1 First Gilmore's bound

The first bound proposed by Gilmore is mainly based on the following observation: Given two vectors x and y in $\Re^n$, their permuted dot (scalar) product is minimized if the elements of vectors are arranged in increasing order for the vector x and in decreasing order for vector y (or vis versa). Using this observation, Gilmore [56] obtained a lower bound for the $\mathcal{QAP}$ by storing the non-diagonal elements of matrices F and D in two vectors say x and y respectively, then minimizes their permuted dot product. For a given partial permutation (or assignment) of facilities α, a lower bound on the assignment π_α representing a completion of the partial permutation α is obtained as follows:

Let b_1 represent a lower bound on the assignment of unassigned facilities, and b_2 a lower bound on the interaction between assigned and unassigned facilities in the partial permutation α. Then b_1 is computed as above by considering now the submatrices F_α and D_α of F and D respectively obtained by excluding rows and columns in α (i.e. rows and columns corresponding to assigned facilities from F and, rows and columns representing their corresponding locations from matrix D). The second part of the lower bound, b_2 is obtained as follows: For each assigned facility i in α, let $f(i,\alpha)$ and $f(\alpha,i)$ be the vectors formed of the elements of row and column i of F respectively, excluding those elements corresponding to facility k in α (in other words for each row or column representing an assigned facility, consider only elements corresponding to unassigned facilities). Similarly, For each assigned

location j in α, let $d(j,\alpha)$ and $d(\alpha,j)$ be the vectors formed of the elements of row and column i of D respectively excluding those elements corresponding to location l in α (in other words for each row or column representing an assigned location, consider only elements corresponding to unassigned locations). Then b_2 is obtained by adding over all elements in α the minimums of the permuted dot products of $f(i,\alpha)$ with $d(j,\alpha)$ and $f(\alpha,i)$ with $d(\alpha,j)$ such that facility i is assigned to location j. A lower bound on π_α is obtained by adding the terms b_1 and b_2 to the exact interaction between assigned facilities in the partial permutation α.

In the case of $\mathcal{GQSAP}$, we modify the computation of such bound by duplicating the elements of D_α, $d(j,\alpha)$ and $f(\alpha,i)$ according to the remaining capacities of the locations for unsaturated ones.

6.2.2 Second Gilmore's bound

The second bound proposed by Gilmore [56], is obtained as follows: For each facility i obtain the vectors $f_r(i)$ and $f_c(i)$ respectively corresponding to row and column i of matrix F excluding the diagonal element. Similarly, for each location j obtain the vectors $d_r(j)$ and $d_c(j)$ respectively corresponding to row and column j of matrix D excluding the diagonal element. For each pair (i,j), let $p_r(i,j)$ and $p_c(i,j)$ represent the permuted products of $f_r(i)$ with $d_r(j)$, and $f_c(i)$ with $d_c(j)$ respectively. Then for $0 \leq \tau \leq 1$, let $\gamma_{ij} = \tau p_r(i,j) + (1-\tau)p_c(i,j)$. The lower bound is obtained

by solving the following linear assignment problem ($\mathcal{LAP}$):

$$\begin{aligned} &\textit{minimize} && \sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}x_{ij} \\ &\textit{Subject to} : \\ & && \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1...n \\ & && \sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1...n \\ & && x_{ij} \in \{0,1\} \end{aligned}$$

Again, for a given partial permutation α, for any pair (i,j) of unassigned facility i and unfilled location j, let $\gamma_{ij} = b_1(i,j) + b_2(i,j)$ such that $b_1(i,j)$ is obtained as above by considering submatrices F_α and D_α of F and D respectively obtained by excluding rows and columns in α (i.e. rows and columns corresponding to assigned facilities from F and, rows and columns representing their corresponding locations from matrix D). The term $b_2(i,j)$ corresponds to the sum of the interactions of element in α with the pair (i,j), that is: $b_2(i,j) = \sum_{m\in\alpha}(f_{im}d_{jl_m} + f_{mi}d_{l_mj})$ where l_m represents the location accommodating facility m under the partial permutation α. As above, a $\mathcal{LAP}$ is solved for the pairs (i,j) of unassigned facilities i and unfilled locations j with coefficients γ_{ij}. A lower bound on π_α is obtained by adding to the solution of the above $\mathcal{LAP}$, the exact interaction between assigned facilities in the partial permutation α.

Again, in the case of $\mathcal{GQSAP}$, the computation of such a bound is obtained by adjusting the elements of D_α, according to the unfilled capacities of the locations for

the unsaturated ones, and solving a linear semi-assignment problem.

6.2.3 Lawler's Bound

A bound equivalent to second Gilmore's bound was obtained by Lawler [86], the Lawler's being more general since it is not restricted to Koopmans and Beckmann problem (**P 1.2**). In this approach, the coefficients of the LAP are obtained by solving the following $\mathcal{LAP}$ for each of the n^2 pairs (i,j) of facility i and location j.

$$\begin{aligned} &minimize && \sum_{k=1}^{n}\sum_{l=1}^{n} d_{ikjl}x_{kl} \\ &Subject\ to\ : && \\ & && \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1...n \\ & && \sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1...n \\ & && x_{ij} \in \{0,1\} \end{aligned}$$

Clearly for the case of the Koopmans and Beckmann problem (**P 1.2**), the coefficient d_{ikjl} may be replaced by the product $f_{ik}d_{jl}$.

In the case of the $\mathcal{GQSAP}$, a similar approach is be used, by solving a semi-assignment problem.

6.3 Starting solutions

6.3.1 A_i-B_j rule

This is a simple and fast construction method proposed by Muller-Merbach [97] to obtain good starting solutions. It is an improved version of the "Best Match" rule that was earlier proposed by Parker [103]. The A_i-B_j rule can be described in the following steps:

At initialization, set M and N as empty sets and start by computing for each $i = 1...n$ and each $j = 1...n$ the quantities A_i and B_j such that

$$A_i = \sum_{p=1}^{n}(f_{ip} + f_{pi}) \qquad B_j = \sum_{p=1}^{n}(d_{jp} + f_{pj})$$

then assign index i with minimal value A_i to an index j with maximal value B_j. Then set $M = M \cup \{i\}$ and $N = N \cup \{j\}$.

At intermediate steps, for $i \notin M$ and $j \notin N$, A_i and B_j are obtained such that

$$A_i = \sum_{p \notin M}(f_{ip} + f_{pi}) \qquad B_j = \sum_{p \notin N}(d_{jp} + f_{pj})$$

then assign index i with minimal value A_i to an index j with maximal value B_j. Then update M and N, and iterate in this fashion until all facilities have been assigned.

In the case of the $\mathcal{GQSAP}$, the selection of facilities and location is done in the same fashion, the only difference is that a location is added to the set N only once it is saturated i.e. cannot accommodate any further facilities.

6.3.2 Increasing degrees of freedom

Also proposed by Muller-Merbach [97], the so-called method of increasing degrees of freedom iteratively updates a partial assignment until a complete permutation of the set $\{1, 2, ..., n\}$ is obtained. The method starts with an empty assignment and a fixed order of the indices $1, 2, ..., n$. Then, given a partial assignment, select the first unassigned index r_k and in a first stage, assign the corresponding facility to an unassigned location and evaluate the increase incurred in the objective function. Repeat the process to all unassigned locations and, each time evaluate the increase incurred. In a second stage, facility r_k is assigned to one of the previously assigned location j_0 and the first stage is performed for its previously corresponding facility i_0 and again the increase in the objective function is evaluated. The process is repeated for all previously assigned facilities. For all $k(n - k + 1)$ possibilities, choose the one with minimum increase in the objective function and use it as the next partial assignment. These steps are repeated until a permutation of the set $\{1, 2, ..., n\}$ is obtained. The method described above, depends on the order of the indices $1, 2, ..., n$. So, to obtain good solutions, the method can be repeated for different orders of the indices $1, 2, ..., n$.

Again for the $\mathcal{GQSAP}$, the described method is easily adjusted such that a location that is not saturated is considered with the free locations as it can accommodate a new facility. On the other hand the interchange that is to be made with assigned facilities in the second stage is done only for those facilities located in a saturated

location since otherwise the possibilities would have been considered in the first stage.

6.4 Algorithm and results

6.4.1 The branch and bound algorithm

In this section we present a branch and bound algorithm for the $\mathcal{GQSAP}$ that uses the Gilmore and the Lawler lower bounds discussed earlier. The algorithm uses the starting methods described above to get an initial upper bound on the solution. Before introducing the algorithm, let us first define the notation that is to be used:

I : The set of assigned facilities.

J : The set of saturated locations (locations that cannot accommodate a new facility).

C^* : The best actual solution.

LB : Lower bound on the current partial assignment.

$P(i)$: The set of prohibited location for facility i. Locations for facility i that would not improve the solution at hand.

Initialization step:

Let: $P(i) = \phi$, $i = 1...n$, $I = \phi$, $J = \phi$.

C^* = value of an assignment obtained using one of the methods for obtaining a starting solution discussed in section 6.3.

X^* = the corresponding assignment.

LB = a lower bound on the actual problem, obtained using one of the methods discussed in section 6.2.

If $LB = C*$ **then**

Go to step 4.

Otherwise

Select a facility $i_1$and place it in location j_1(this selection is done based on a criterion similar to the A_i-B_j rule discussed above).

Set $I = \{i_1\}$, $n_{j_1} = n_{j_1} - 1$. **If** $n_{j_1} = 0$ **then** $J = \{j_1\}$ **end if.**

Set $k = 1$

Go to step 1.

End if

Step 1:[Forward step]

compute LB a lower on the completion of the current partial permutation [taking into consideration the assignment already made and setting $\gamma_{ij} = \infty$ $\forall i = 1...n$ and $j \in P(i)$]

If $LB > C^*$ **then**

Go To step 2.

Otherwise

Select a facility $i_{k+1} \notin I$ and place it in location $j_{k+1} \notin J \cup P(i_{k+1})$

Set $I = I \cup \{i_{k+1}\}$, $n_{j_{k+1}} = n_{j_{k+1}} - 1$. **If** $n_{j_{k+1}} = 0$ **then** $J = J \cup \{j_{k+1}\}$ **end if.**

Set $k = k + 1$

If $k = m$ **then**

We have a complete assignment. Compute its cost C.

If $C < C^*$ **then**

$C^* = C$, $X^* = X$

End if

Go to step 2.

Otherwise

Go to step 1.

End if

Step 2: [partial fathoming]

Set $I = I - \{i_k\}$, $J = J - \{j_k\}$, $n_{j_k} = n_{j_k} + 1$. $P(i_k) = P(i_k) \cup \{j_k\}$.

Compute LB a lower on the completion of the current partial permutation

If $LB \geq C^*$ **then**

Go to step 3.

Otherwise

Assign i_k to some location $j_k \notin J \cup P(i_{k+1})$

Set $I = I \cup \{i_k\}$, $n_{j_k} = n_{j_k} - 1$. **If** $n_{j_k} = 0$ **then** $J = J \cup \{j_k\}$ **end if.**

Go to step 1.

End if

Step 3: [backtracking]

Set $I = I - \{i_k\}$, $J = J - \{j_k\}$, $n_{j_k} = n_{j_k} + 1$. $P(i_k) = \phi$

Set $k = k - 1$

If $k = 0$ **then**

Go to step 4

Otherwise

Go to step 2

End if

Step 4:

Stop. The optimal solution is the permutation stored in X^* with optimal cost C^*.

6.4.2 The results

The methods described above have been tested on some problems whose description is given in Table 6.1. The problems are obtained from the classical Nugent et al. [98] set of problems. The Nugent problems consider adjacent locations on a two dimensional gird, with rectilinear distances. These problems are transformed such that the locations on each row are grouped into one single location with capacity equal to the number of locations grouped. The distances between any two of the newly created locations are defined to be the maximum of the distance between two locations allocated to the two new locations. The flow between facilities is the same as in the case of the Nugent problems. These problems are new and may be used as benchmarks in future study of $\mathcal{GQSAP}$. Table 6.1 gives a summary of the characteristics of these test problems.

Problem	n	m	capacities	Optimal
P1	5	2	2,3	42
P2	6	2	3,3	54
P3	7	3	1,3,3	166
P4	8	2	4,4	244
P5	12	3	4,4,4	698
P6	14	3	5,5,4	1332
P7	15	3	5,5,5	1604
P8	16	4	5,5,5,1	2336
P9	16	4	4,4,4,4	1772
P10	17	4	5,5,5,2	2610
P11	18	4	5,5,5,3	2988
P12	20	4	5,5,5,5	3956
P13	21	3	7,7,7	3854
P14	22	2	11,11	4950

Table 6.1: Characteristics of the test problems

Table 6.2 gives the results obtained for the two methods used to obtain a starting feasible solution, namely the method of increasing degrees of freedom and the A_i-B_i technique, the percentage deviation of the solution of the two methods from the optimal solution is reported for each problem.

Pb	Optimal	A_i-B_i	% Dev	Inc. D. F	% Dev
P1	42	54	28.6	42	0
P2	54	54	0	54	0
P3	166	166	0	170	2.4
P4	224	272	21.4	224	0
P5	698	846	21.2	730	4.6
P6	1332	1526	14.6	1340	0.6
P7	1604	1966	22.6	1610	0.4
P8	2336	2698	15.5	2514	7.6
P9	1772	1984	12.0	1944	9.7
P10	2610	3084	18.2	2758	5.7
P11	2988	3284	9.9	3154	5.6
P12	3956	4144	4.8	4172	5.5
P13	3854	4146	7.6	3962	2.8
P14	4950	5280	6.7	5192	4.9

Table 6.2: Strting solutions and percentage deviation from optimal

From Table 6.2 it is clearly seen that the method of increasing degrees of freedom, whose deviation is within 10% from the optimal solution, performs better that the A_i-B_i rule which deviates in some cases to almost 30%.

Table 6.3 gives the value of the lower bounds obtained using the first and second Gilmore bounds.

Pb	Optimal	1^{st} Gilmore Bound		2^{nd} Gilmore Bound	
		Bound	% Dev.	Bound	% Dev.
P1	42	30	28.57	30	28.57
P2	54	42	22.22	42	22.22
P3	166	114	31.33	123	25.9
P4	224	80	64.29	92	62.3
P5	698	464	33.52	488	30.09
P6	1332	852	36.04	890	33.18
P7	1604	1060	33.92	1116	30.42
P8	2336	1686	27.83	1733	25.81
P9	1742	1216	30.2	1233	29.22
P10	2610	1880	27.97	1896	27.36
P11	2988	2190	26.71	2197	26.47
P12	3956	2852	27.91	2876	27.3
P13	3854	1890	50.96	2227	42.22
P14	4950	1342	72.89	1617	67.33

Table 6.3: Lower bounds obtained using 1^{st} and 2^{nd} Gilmore bounds

Remark 9 *It is to be noted that the Gilmore-Lawler bound, obtained by solving successive assignment problems, and the second Gilmore bound, where only a single assignment is solved, perform identically in terms of the value of the lower bound and the number of iterations, thus we reported only results for the second Gilmore's bound. The difference can be seen when computation times are reported.*

We see that the two lower bounds are weak, although the second bound is tighter than the first one as it gives closer values to the optimum than the first bound. This was expected since for the $\mathcal{QAP}$ the bounds behave in a similar way. On the other hand, we can observe that the bounds are not affected (or very little) by the size of the problem which is not the case for the $\mathcal{QAP}$.

Table 6.4 gives number of iterations needed to reach optimality using the different lower bounds and the two starting solutions.

	1^{st} Gilmore		2^{nd} Gilmore	
Pb	A_i-B_i **Rule**	**I. D. F.**	A_i-B_i **Rule**	**I. D. F.**
P1	9	0	5	0
P2	0	0	0	0
P3	0	26	0	21
P4	5	0	5	0
P5	553	553	143	143
P6	12247	12247	1833	1833
P7	74	56	38	22
P8	75735	75117	6883	6665
P9	29702	29684	3050	3036
P10	202698	202698	10964	10964
P11	734058	732848	28194	27950
P12	602741	602757	20241	20259
P13	4369	4333	733	715
P14	32	32	32	32

Table 6.4: Number of iterations needed to reach optimality prior to stopping

It can be observed here that the starting solution does not have a significant effect on the number of iteration needed to reach optimality. But clearly the second bound performs significantly better than the first one.

Table 6.5 gives the time needed to reach optimality using the different Lower bounds together with the two starting solutions.

	1^{st} Gilmore		2^{nd} Gilmore		Gilmore-Lawler	
Pb	**A_i-B_i**	**I. D. F.**	**A_i-B_i**	**Inc. D. F**	**A_i-B_i**	**I. D. F.**
P1	0	0	0	0	0	0
P2	0	0	0	0	0	0
P3	0	0	0	0	0	0.05
P4	0	0	0	0	0	0
P5	0.17	0.16	0.06	0.11	0.55	0.55
P6	4.01	4.06	0.93	0.99	9.23	9.23
P7	0.06	0.06	0.06	0.06	0.22	0.26
P8	35.53	35.60	5.38	5.38	80.35	78.76
P9	12.58	12.64	1.98	2.03	33.12	33.50
P10	103.64	104.03	10.05	10.16	143.79	144.68
P11	399.26	395.19	25.71	25.76	411.94	411.39
P12	430.83	373.54	18.57	18.73	353.89	358.17
P13	150.64	141.86	0.55	0.82	7.14	7.35
P14	0.06	0.33	0.06	0.33	0.11	0.38

Table 6.5: Time needed to reach optimality

When time is of concern, we see that the second bound performs better than the other two bounds. This difference is due to the fact that the first bound is weaker, on thee other hand the third bound requires more time due to high number of semi-assignment problems needed to be solved. Again, the effect of the starting solution is almost insignificant.

Table 6.6 reports the total number of iterations and the number of branches made using the different lower bounds with the two starting solutions.

	1^{st} Gilmore Bound				2^{nd} Gilmore Bound			
	A_i-B_i		I. D. F.		A_i-B_i		I. D. F.	
Pb	Iter	Bran	Iter	Bran	Iter	Bran	Iter	Bran
P1	11	5	7	3	7	3	5	2
P2	11	5	11	5	11	5	11	5
P3	36	14	38	15	24	9	28	11
P4	71	35	71	35	27	13	27	13
P5	1305	456	1305	456	287	102	287	102
P6	16833	6175	16833	6175	2416	887	2416	887
P7	15200	5395	15182	5386	1962	711	1946	703
P8	105863	33521	105245	33233	9169	2826	8951	2727
P9	335980	9775	335962	97726	24017	6808	24003	6801
P10	505635	147371	505635	147371	26014	7216	26014	7216
P11	2115797	613250	2114587	612752	72238	20552	71994	20459
P12	18708029	5308096	18708029	5308104	324172	92791	324190	92800
P13	5925730	2010551	5925694	2010533	137600	47611	137582	47602
P14	104347	52173	104347	52173	6141	3070	6141	3070

Table 6.6: Total number of iterations and branches of the branch & bound algorithms

Following the same pattern, we see that the second bound performs better than the first one. Here again the starting solution has almost no effect.

Finally, table 6.7 compares the total computation times for the different bounds and the different starting solutions.

Pb	1st Gilmore Bound		2nd Gilmore Bound		Gilmore-Lawler	
	A_i-B_i	I. D. F.	A_i-B_i	I. D. F.	A_i-B_i	I. D. F.
P1	0	0	0	0	0	0
P2	0	0	0	0	0	0
P3	0	0	0	0	0.06	0.05
P4	0	0	0	0	0	0.05
P5	0.39	0.38	0.17	0.17	1.05	1.10
P6	5.61	5.65	1.32	1.32	12.19	12.20
P7	6.49	6.54	1.43	1.43	11.37	11.31
P8	50.69	50.76	7.47	7.47	107.65	106.12
P9	38.62	38.68	19.50	19.50	270.95	274.18
P10	269.19	269.63	25.71	25.87	343.61	346.69
P11	1202.32	601.28	72.50	72.56	1068.35	1157.50
P12	13618.99	13590.93	397.72	337.27	7853.59	8843.60
P13	6127.12	5329.44	162.58	162.09	1752.18	1440.75
P14	76.07	76.46	7.09	7.31	31.20	31.47

Table 6.7: Total running time (in seconds) of the branch and bound algorithms

In terms of running time also, we see that the second bound out-performs the two other bounds, which leaves no doubt on which method should be recommended.. Again the insignificance of the effect of the starting solution on the total running time can be seen here.

6.5 Conclusion

From the above results, we see that the computation time needed for the second starting method of solution (increasing degrees of freedom) is relatively high on the average (16 %) compared to the time needed for the A_i-B_i technique, yet its contribution in reducing the total number of iterations is relatively insignificant. Thus we suggest to use the A_i-B_i technique. On the other hand, as it was expected, the first Gilmore's bound is weaker than the other lower bounds, thus resulting in a very high computation time. The two other lower bounds perform similarly, yet the time needed to compute the Gilmore-Lawler bound is very high compared to the second Gilmore's bound, this is attributed to the time needed to solve the $(nm + 1)$ $\mathcal{LAP}$. Hence we suggest the use of the second Gilmore's bound (where a single $\mathcal{LAP}$ is solved), with the A_i-B_i technique as a starting upper bound.

Chapter 7

Relaxation Linearization Technique

7.1 Introduction to $\mathcal{RLT}$

Recently, a new technique the so-called Relaxation Linearization Technique, or $\mathcal{RLT}$ for short, has been introduced by Sherali and Adams [123]. The technique is meant for treatment of discrete and continuous nonconvex programming problems. In an automatic reformulation way, strong valid constraints are generated to end up with a tight linear representation of the original problem. The procedure is capable of generating a hierarchy of representations of increasing degree of strength at a cost of increasing problem size. In our case, we are rather interested in the applications of the technique to discrete problems and in particular to the $\mathcal{QAP}$ and $\mathcal{GQAP}$. In the case of integer 0-1 problems, the technique tries to obtain a formulation with a relaxation that closely approximate the convex hull of the original integer program.

To obtain the first level in the hierarchy, the $\mathcal{RLT}$ operates in two phases, in the first phase or reformulation phase, the constraints of the problem are multiplied by the integer variables and their complements (i.e. x and $1-x$ where x is a 0-1 variable of the original problem), duplicating all constraints for each of the variables of the problem. In the second phase or linearization phase, new variables corresponding to the products generated are introduced. The nonlinearity is taken out from the generated problem in the first phase, by replacing the products with the newly introduced variables.

A higher level d where $2 \leq d \leq n$ in the hierarchy is obtained by multiplying the constraints by polynomials of degree d involving the n binary variables and their complements. For each level $d \in \{1, 2, ..., n\}$, the feasible region is lifted to a higher dimensional representation of the feasible region in terms of the original variables of the problem an the newly introduced variables.

7.2 $\mathcal{RLT}$ applied to the $\mathcal{QAP}$

In this section we apply the previously discussed two phases of the $\mathcal{RLT}$ technique to the quadratic assignment problem. The discussion will be restricted to the first level of the technique only. Higher levels may be applied by following the same steps. This would lead to tighter lower bounds at a cost of increase in problem sizes.*

7.2.1 Relaxation-Linearization

In the first phase, the $2n$ constraints of the problem that are of the form

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

and

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

are multiplied by the n^2 variables and their complements, obtaining for each variable x_{kl} $k, l = 1...n$ equalities of the form

$$\sum_{i=1}^{n} x_{kl}x_{ij} = x_{kl} \qquad j = 1...n$$
$$\sum_{i=1}^{n} (1 - x_{kl})x_{ij} = 1 - x_{kl} \qquad j = 1...n$$

and

$$\sum_{j=1}^{n} x_{kl}x_{ij} = x_{kl} \qquad i = 1...n$$
$$\sum_{j=1}^{n} (1 - x_{kl})x_{ij} = 1 - x_{kl} \qquad i = 1...n$$

If we reorder the second type of equalities, obtained by multiplying the original equalities by $(1 - x_{kl})$, we get

$$\begin{aligned}
& \sum_{i=1}^{n} (x_{ij} - x_{kl}x_{ij}) = 1 - x_{kl} \\
\Leftrightarrow \quad & \sum_{i=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{kl}x_{ij} = 1 - x_{kl} \\
\Leftrightarrow \quad & 1 - \sum_{i=1}^{n} x_{kl}x_{ij} = 1 - x_{kl} \qquad \text{since } \sum_{i=1}^{n} x_{ij} = 1 \\
\Leftrightarrow \quad & \sum_{i=1}^{n} x_{kl}x_{ij} = x_{kl}
\end{aligned}$$

The same argument holds when we sum up over the second index j. So, the second type of equalities (those obtained by multiplying by $1 - x$) are redundant and hence should be dropped from consideration.

On the other hand, for a given index k, the equality $\sum_{j=1}^{n} x_{kl}x_{kj} = x_{kl}$ will have on the left hand side the term x_{kl}^2 appearing, and since $x_{kl} \in \{0, 1\}$, then $x_{kl}^2 = x_{kl}$, leaving us with $\sum_{j=1, j\neq l}^{n} x_{kl}x_{kj} = 0$. Such products do not appear in the $\mathcal{QAP}$ objective function, in addition to the fact that they imply $x_{kl}x_{kj} = 0$, since $x_{ij} \geq 0\ \forall i, j = 1...n$. Hence such equalities should be taken out from consideration. Now, considering the remaining equalities together with the original problem constraints, we end up with the following integer nonlinear problem that is equivalent to the original $\mathcal{QAP}$.

P 7.1

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}x_{ij}x_{kl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k\neq i}}^{n} x_{ij}x_{kl} = x_{kl} \qquad i, j, l = 1...n, j \neq l$$

$$\sum_{\substack{j=1 \\ l\neq j}}^{n} x_{ij}x_{kl} = x_{kl} \qquad i, j, k = 1...n, i \neq k$$

$$x_{ij}x_{kl} = x_{kl}x_{ij} \qquad i, j, k, l = 1...n, i < k, l \neq j$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1...n$$

In the second phase, we add the new variables y_{ijkl} to replace the products $x_{ij}x_{kl}$,

and substituting in the original problem, we end up with the following problem:

P 7.2

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k\neq i}}^{n} y_{ijkl} = x_{kl} \qquad i,j,l = 1...n, j \neq l$$

$$\sum_{\substack{j=1 \\ l\neq j}}^{n} y_{ijkl} = x_{kl} \qquad i,j,k = 1...n, i \neq k$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l = 1...n, i < k, l \neq j$$

$$x_{ij}, y_{ijkl} \in \{0,1\} \qquad i,j,k,l = 1...n, k \neq i, j \neq l$$

It can be observed that integrality of x_{ij} imply integrality of y_{ijkl}, and that y_{ijkl} are automatically upper bounded by 1 from the assignment constraints. Hence we have the following equivalent formulation

P 7.3

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k\neq i}}^{n} y_{ijkl} = x_{kl} \qquad i,j,l = 1...n, j \neq l$$

$$\sum_{\substack{j=1 \\ l \neq j}}^{n} y_{ijkl} = x_{kl} \qquad i,j,k = 1...n, i \neq k$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l = 1...n, i < k, l \neq j$$

$$x_{ij} \in \{0,1\}, y_{ijkl} \geq 0 \qquad i,j,k,l = 1...n, k \neq i, j \neq l$$

This formulation corresponds to the formulation given by Adams and Johnson (**P 2.10**). It can be noted that the last set of equalities can be substituted directly in the other constraints reducing the number of constraints and variables by a factor of the order of n^4.

7.2.2 Inequality constraints

A relaxation of the above problem (**P 7.3**) given by

P 7.4

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k \neq i}}^{n} y_{ijkl} = x_{kl} \qquad i,j,l = 1...n, j \neq l$$

$$\sum_{\substack{j=1 \\ l \neq j}}^{n} y_{ijkl} = x_{kl} \qquad i,j,k = 1...n, i \neq j$$

$$y_{ijkl} = y_{ijkl} \qquad i,j,k,l = 1...n, i < k, l \neq j$$

$$0 \leq x_{ij} \leq 1, y_{ijkl} \geq 0 \qquad i,j,k,l = 1...n, k \neq i, j \neq l$$

have been used in a branch and bound scheme using an interior point algorithm to solve the resulting linear programs by Resende et al. [112] and provided very encouraging results. We propose here to further strengthen their relaxation by adding inequality constraints to obtain an even tighter relaxation. We can observe that one of the problems with above relaxation is that it may happen that the following products may not be satisfied:

$$x_{ij}x_{kl} = y_{ijkl}$$

We try to fulfill such constraints by adding the following inequalities to the problem

$$\begin{aligned} y_{ijkl} &\leq x_{ij} \\ y_{ijkl} &\leq x_{kl} \\ y_{ijkl} &\geq x_{ij} + x_{kl} - 1 \end{aligned}$$

It is clear that if x_{ij} or x_{kl} (or for that matter both) is 0, then the first two inequalities insure that y_{ijkl} is 0, on the other hand if both x_{ij} and x_{kl} are 1 than the third inequality will force y_{ijkl} to be 1. On the other hand, the first two inequalities are directly implied by equalities of the form

$$\sum_{\substack{i=1 \\ k \neq i}}^{n} y_{ijkl} = x_{kl}$$

Hence we only need to add the third set of inequalities to the above relaxation. To get the following problem:

P 7.5

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1...n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k\neq i}}^{n} y_{ijkl} = x_{kl} \qquad i,j,l = 1...n, j < l$$

$$\sum_{\substack{j=1 \\ l\neq j}}^{n} y_{ijkl} = x_{kl} \qquad i,j,k = 1...n, i < k$$

$$y_{ijkl} \geq x_{ij} + x_{kl} - 1 \qquad i,j,k,l = 1...n, i < k, l \neq j$$

$$0 \leq x_{ij} \leq 1, y_{ijkl} \geq 0 \qquad i,j,k,l = 1...n, k > i, j \neq l$$

Theorem 9 *The relaxation given in (**P 7.5**) is at least as tight as the Adams and Johnson's relaxation given in (**P 2.10**).*

Proof. The proof is strait forward. Since imposing integrality on the x_{ij} variables would give us an equivalent problem to the $\mathcal{QAP}$ this is because the Adams and Johnson's formulation (**P 2.10**) is proven to be equivalent to the $\mathcal{QAP}$ (see [1]), and we only added the inequality constraints that do not disturb the equivalence at all. On the other hand in the relaxation we are adding the inequality constraints that would only make the solution to the relaxation in (**P 7.5**) greater than the relaxation of Adams and Johnson (**P 2.10**), making the relaxation in (**P 7.5**) tighter. ■

Remark 10 *During testing it turned out that the lower bound obtained by imposing the inequality constraints does not improve the lower bound although it was noted that the linear problems seemed to be solved faster in terms of time.*

7.3 $\mathcal{RLT}$ applied to $\mathcal{GQSAP}$

In an identical way as done in the previous section, we apply the two phases of the $\mathcal{RLT}$ technique to the generalized quadratic semi-assignment problem. In this case also the discussion will be restricted to the first level of the technique only, as higher level would generate a huge number of variables and constraints that would be very difficult to handle.

7.3.1 Linearization

In a similar spirit as it was done in the previous section, the first phase of the $\mathcal{RLT}$ technique is applied to the $n + m$ constraints of the problem that are of the form

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1...n$$

and

$$\sum_{i=1}^{n} x_{ij} = n_j \qquad j = 1, ..., m$$

Multiplying these constraints by the $n \times m$ variables of the problem and their complements, will result in equalities of the following form, for each variable x_{kl}

such that $k = 1...n$ and $l = 1, ..., m$

$$\sum_{j=1}^{m} x_{kl}x_{ij} = x_{kl} \qquad i = 1...n$$
$$\sum_{j=1}^{m} (1 - x_{kl})x_{ij} = 1 - x_{kl} \qquad i = 1...n$$

and

$$\sum_{i=1}^{n} x_{kl}x_{ij} = n_j x_{kl} \qquad j = 1, ..., m$$
$$\sum_{i=1}^{n} (1 - x_{kl})x_{ij} = n_j(1 - x_{kl}) \qquad j = 1, ..., m$$

Reordering the second type of equalities, those obtained by multiplying the original equalities by $(1 - x_{kl})$, for the first set of equalities (where the summation is done over the second index representing locations) we get the same derivations obtained in the previous section. For the second set (where the summation is done over the first index representing facilities)

$$\begin{aligned}
& \sum_{i=1}^{n} (x_{ij} - x_{kl}x_{ij}) = n_j(1 - x_{kl}) \\
\Leftrightarrow \quad & \sum_{i=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{kl}x_{ij} = n_j - n_j x_{kl} \\
\Leftrightarrow \quad & n_j - \sum_{i=1}^{n} x_{kl}x_{ij} = n_j - n_j x_{kl} \qquad \text{since } \sum_{i=1}^{n} x_{ij} = n_j \\
\Leftrightarrow \quad & \sum_{i=1}^{n} x_{kl}x_{ij} = n_j x_{kl}
\end{aligned}$$

So, the second type of equalities (those obtained by multiplying by $1 - x$) are redundant and hence should be dropped from consideration.

On the other hand, for a given index k,the term x_{kl}^2 will appear in both types of equalities, and since $x_{kl} \in \{0, 1\}$, then $x_{kl}^2 = x_{kl}$.

This lead to the first set of equalities $\sum_{j=1,j\neq l}^{m} x_{kl}x_{kj} = 0$ being unnecessary as seen above.

The second type of equalities would lead to $\sum_{i=1,i\neq k}^{n} x_{kl}x_{il} = (n_l - 1)x_{kl}$. Now, considering the remaining equalities together with the original problem constraints, we end up with the following integer nonlinear problem that is equivalent to the original $\mathcal{GQSAP}$.

P 7.6

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}x_{ij}x_{kl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = n_j \qquad j = 1...m$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1\\k\neq i}}^{n} x_{ij}x_{kl} = (n_j - 1)x_{kl} \qquad 1 \leq i \leq n, 1 \leq j, l \leq m, j \neq l$$

$$\sum_{\substack{j=1\\l\neq j}}^{m} x_{ij}x_{kl} = x_{kl} \qquad 1 \leq i, k \leq n, 1 \leq j \leq m, k \neq i$$

$$x_{ij}x_{kl} = x_{kl}x_{ij} \qquad i, k = 1...n, j, l = 1...m, i < k, l \neq j$$

$$x_{ij} \in \{0, 1\} \qquad i = 1...n, j = 1...m$$

In the second phase, we add the new variables y_{ijkl} to replace the products $x_{ij}x_{kl}$, and substituting in the original problem, we end up with the following problem:

P 7.7

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = n_j \qquad j = 1, ..., m$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k\neq i}}^{n} y_{ijkl} = (n_j - 1)x_{kl} \qquad 1 \leq i \leq n, 1 \leq j, l \leq m, j \neq l$$

$$\sum_{\substack{j=1 \\ l\neq j}}^{m} y_{ijkl} = x_{kl} \qquad 1 \leq i, k \leq n, 1 \leq j \leq m, k \neq i$$

$$y_{ijkl} = y_{klij} \qquad 1 \leq i, k \leq n, 1 \leq j, l \leq m, i < k, l \neq j$$

$$x_{ij}, y_{ijkl} \in \{0, 1\} \qquad 1 \leq i, k \leq n, 1 \leq j, l \leq m, k \neq i, j \neq l$$

It can be observed that integrality of x_{ij} imply integrality of y_{ijkl}, and that y_{ijkl} are automatically upper bounded by 1 from the first semi-assignment constraints. Taking this into consideration can contribute in relaxing further the problem.

Hence we have the following equivalent formulation

P 7.8

$$\textit{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} f_{ik}d_{jl}y_{ijkl}$$

subject to :

$$\sum_{i=1}^{n} x_{ij} = n_j \qquad j = 1, ..., m$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1...n$$

$$\sum_{\substack{i=1 \\ k\neq i}}^{n} y_{ijkl} = (n_j - 1)x_{kl} \qquad 1 \leq i \leq n, 1 \leq j, l \leq m, j \neq l$$

$$\sum_{\substack{j=1 \\ l\neq j}}^{m} y_{ijkl} = x_{kl} \qquad 1 \leq i, k \leq n, 1 \leq j \leq m, k \neq i$$

$$y_{ijkl} = y_{klij} \qquad 1 \leq i, k \leq n, 1 \leq j, l \leq m, i < k, l \neq j$$

$$x_{ij} \in \{0, 1\}, y_{ijkl} \geq 0 \qquad 1 \leq i, k \leq n, 1 \leq j, l \leq m, k \neq i, j \neq l$$

It can be noted that the last set of equalities can be substituted directly in the other constraints reducing extensively the number of variables and constraints.

Theorem 10 *The above problem (**P 7.8**) is equivalent to the original $\mathcal{GQSAP}$.*

7.4 The Algorithm

The same notation as in the previous chapter will be adopted in this chapter. The branch and bound algorithm for this case is shown below:

Initialization step:

Let: C^* = value of an assignment obtained using one of the starting solutions methods discussed in chapter 6.

X^* = the corresponding assignment.

LB = a lower bound on the actual problem, obtained by solving the relaxation of problem (**P 7.8**).

If $LB = C^*$ **then**

Go to step 4.

Otherwise

Set $k = 1$, $Stack = \emptyset$

Go to step 1.

End if

Step 1:[Forward step]

Select an $x_{i_k j_k}$with maximum value such that $0 < x_{i_k j_k} < 1$.

Set $x_{i_k j_k} = 1$, $indic(k) = 1$

Set $Stack = Stack \cup \{x_{i_k j_k}\}$

Compute LB a lower bound on the current LP (that is (**P 7.8**) and the constraints $x_{i_k j_k} = 1$ or 0).

If solution is integer **then**

If $LB < C^*$ **then** $C^* = LB$, Store solution in X^* and Go To step 2 **end if.**

Otherwise

If $LB \geq C^*$ **then**

Go To step 2.

Otherwise

Go to step 1.

End if

Step 2: [partial fathoming]

Set $x_{i_k j_k} = 0$, $indic(k) = 0$

Compute LB a lower bound on the actual problem.

If $LB \geq C^*$ **then**

Go to step 3.

Otherwise

Go to step 1.

End if

Step 3: [backtracking]

Set $Stack = Stack - \{x_{i_k j_k}\}$

Set $k = k - 1$

If $k = 0$ **then**

Go to step 4

Otherwise

Go to step 2

End if

Step 4:

Stop. The optimal solution is the solution stored in X^* with optimal cost C^*.

7.5 Summary of the results

In the following table we summarize the results obtained when applying the $\mathcal{RLT}$ to $\mathcal{GQSAP}$. We also provide comparison with the best results obtained from the previous chapter.

Pb	Optimal	RLT Bound		2^{nd} Gilmore Bound	
		Bound	% Dev.	Bound	% Dev.
P1	42	42	0	30	28.57
P2	54	54	0	42	22.22
P3	166	145	12.65	123	25.90
P4	224	128	42.54	92	62.30
P5	698	572	18.05	488	30.09
P6	1332	1015	23.80	890	33.18
P7	1604	1216	24.19	1116	30.42
P8	2336	1895	18.88	1733	25.81
P9	1742	1362	23.16	1233	29.22
P10	2610	2105	19.35	1896	27.36
P11	2988	2378	20.41	2197	26.47
P12	3956	3061	22.62	2876	27.3
P13	3854	2567	33.39	2227	42.22
P14	4950	1936	60.899	1617	67.33

Table 7.1: Lower bounds obtained usig RLT and Gilmore

From the results shown in table 7.1 it is clear that the $\mathcal{RLT}$ bound outperforms the best of Gilmore bounds. In fact the average percent deviation for the $\mathcal{RLT}$ does

not exceed 23.8% while the Gilmore's is around 32%. Hence there is no doubt that the $\mathcal{RLT}$ bound is tighter which was expected.

Table 7.2 gives number of iterations and the time in seconds needed to reach optimality using the two lower bounds

	RLT		2^{nd} Gilmore	
Pb	Time (sec)	Iterations	Time (sec)	Iterations
P1	0	0	0	0
P2	0	0	0	0
P3	0.16	6	0	21
P4	0	0	0	0
P5	29.33	24	0.11	143
P6	292.04	77	0.99	1833
P7	485.32	82	0.06	22

Table 7.2: Number of iterations and time needed to reach optimality prior to stopping

It can be observed here the $\mathcal{RLT}$ again outperforms the other method in terms of number of iterations to reach optimality. This is again expected since the $\mathcal{RLT}$ bound is tighter. On the other hand the time required to reach optimality is by far higher in the case of the $\mathcal{RLT}$. This due to the fact that a very large Linear Program need to be solved at each branch which is very expensive in terms of time..

Table 7.3 reports the total number of iterations, the total number of branches made and the total running time using the different lower bounds needed to terminate the algorithms.

	RLT			2^{nd} Gilmore Bound		
Pb	**Iter**	**Bran**	**Time (sec)**	**Iter**	**Bran**	**Time (sec)**
P1	0	0	0	5	2	0
P2	0	0	0	11	5	0
P3	8	3	0.22	28	11	0
P4	7	3	0.06	27	13	0
P5	33	15	45.31	287	102	0.38
P6	226	112	827.12	2416	887	5.65
P7	334	116	1389.29	1946	703	6.54

Table 7.3: Total number of iterations, branches and running timeof the branch bound algorithms

Again here, it is clear that in terms of number of iterations and number of branches the $\mathcal{RLT}$ based algorithm performs better. Unfortunately it is not the case when the performance measure is the time. This fact is mainly due to the expensiveness of the $\mathcal{RLT}$ lower bound

7.6 Conclusion

In this chapter we presented a technique due to Sherali and Adams [123] namely the Relaxation Linearization Technique or $\mathcal{RLT}$. This technique was applied to $\mathcal{GQSAP}$ and computational results were presented. A comparison of this technique with a previously discussed method (based on Gilmore's bound) shows the thightness of the $\mathcal{RLT}$ based bound but due to its high cost in terms of computation time, which is indeed the major performance measure in practice, we do not recommend its use unless faster ways to obtain such bounds are devised. Although we should stress that better branching procedures might improve the performance of such algorithm taking advantage of the thightness of the $\mathcal{RLT}$ bound.

Chapter 8

Conclusion and Further research

8.1 Conclusion

The Quadratic Assignment Problem is one of the hardest combinatorial optimization problems. It has a wide range of applications in the areas of computer science and operations research. In this dissertation, the $\mathcal{QAP}$ has been studied further, its formulation is generalized and various approaches for solving it and a class of its extensions have been proposed.

The first approach proposed in this dissertation is done by examining the network structure of the problem and using a network simplex like pivot to obtain a characterization of local star minima with respect to a given basis. An algorithm for finding a B-local minimum using such a characterization was also proposed.

In a second approach, the $\mathcal{QAP}$ is reformulated as what is called a variable net-

work. The properties of the new formulation were investigated. Furthermore, ways to use this new formulation in designing algorithms for the $\mathcal{QAP}$ were discussed.

It was noticed that the formulation of the $\mathcal{QAP}$ does not capture the general case of the facility location problem. Therefore a generalized formulation was proposed. The generalization of the $\mathcal{QAP}$ was called " *The Generalized Quadratic Assignment Problem* or $\mathcal{GQAP}$ ". The $\mathcal{GQAP}$ was shown to have the $\mathcal{QAP}$ and all its derivatives as special cases.

A special class of the so-called $\mathcal{GQAP}$ was investigated and several branch and bound algorithms to solve it were designed. The lower bounds for the algorithms proposed were based on an adaptation of the well known Gilmore bound [56] and a lower bound obtained using the $\mathcal{RLT}$ technique developed by Sherali and Adams [123] for general mixed integer programming problems.

The results showed that, although the bound obtained using $\mathcal{RLT}$ was very tight and outperformed by far the Gilmore's based bound, it was very costly in terms of computation time. This high running time was mainly due to the large size of the linear program that is solved to obtain the lower bound. Hence the Gilmore's bound based branch and bound algorithm performed better than the $\mathcal{RLT}$ based branch and bound algorithm, yet the Gilmore's based lower bound was weaker.

To take advantage of the tight bound obtained by the $\mathcal{RLT}$ approach, suggestions on that aspect are included in the next section.

8.2 Further Research

The topics investigated in this dissertation open new horizons for further research. In this context, we observe that the characterization proposed in chapter 3 could be used in designing new algorithms or Meta-heuristics such as Genetic algorithms or Tabu search for the $\mathcal{QAP}$.

On the other hand, the reformulation proposed in chapter 4 could be used in designing branch and bound algorithms for the $\mathcal{QAP}$. It is to be observed that although a bound based on such formulation would be weaker that the $\mathcal{RLT}$ based bounds the constraints in that formulation represent aggregations of the $\mathcal{RLT}$ constraints, nevertheless it would be worth investigating since the problem size in the $\mathcal{RLT}$ is by far larger. Thus getting a lower bound based on the proposed formulation would be less costly.

The generalized formulation, proposed in chapter 5, offers a new class of problems that need efficient algorithms to solve. Further research need to be done in developing effective algorithms for the $\mathcal{GQAP}$ and some of its special cases.

The characteristics of the polytope of the generalized problem need to be investigated to see its properties.

Another topic that could be investigated is the selection of the branching variables. A better strategy for such selection is definitely needed in order to improve the performance of the branch and bound algorithm that uses the $\mathcal{RLT}$ based lower bound. Combining the second Gilmore's bound and the $\mathcal{RLT}$ based bound may turn

out to be very efficient for solving $\mathcal{GQSAP}$ by taking advantage of the Gilmore's in early nodes of the branch and bound tree, and in later nodes where a stronger bound is needed, the $\mathcal{RLT}$ bound is used. In this case the $\mathcal{RLT}$ problem would be smaller since many x_{ij} variables would have been fixed hence no need to generate $\mathcal{RLT}$ cuts with respect to these variables.

The $\mathcal{GQSAP}$ may be even more useful in solving the actual $\mathcal{QAP}$. This could be done in some decomposition form. In a first stage, group the locations that are neighbors into a single location with higher capacity and solve the resulting $\mathcal{GQSAP}$. In a second Stage of the decomposition solve a number of smaller $\mathcal{QAP}$'s that would be easier to handle than the original one.

Appendix A

```
C   Main program. Reads the data of the problem.
C
C   The dimensions of the problem:
C   N   : The number of facilities to be assigned.
C   M   : The number of locations to accomodate the facilities
C   MF  : An N*N matrix representing the flows between facilities
C   MD  : An M*M matrix representing the distances between locations
C   MC  : An N*M matrix representing the cost of assigning a facility to a
C         location. This represents the linear term of the objective
function
C   LC  : A vector of capacity M representing the location capacities
C
      parameter (ndim=100,mdim=100)
      implicit integer (a-z)
      Dimension mf(ndim,ndim),md(mdim,mdim),mc(ndim,mdim),lc(mdim)
      open(1,file='data.in')

c
c ndom and Nran correspondand to the domain and range of a partial
assignment
c idom number of assigned facilities
c
      read(1,*)n
      read(1,*)m
      do i=1,n
      read(1,*)(mf(i,j),j=1,n)
      enddo
      do i=1,m
      read(1,*)(md(i,j),j=1,m)
      enddo
      read(1,*)nl
      if(nl.eq.1)then
      do i=1,n
      read(1,*)(mc(i,j),j=1,m)
      enddo
      endif
      do i=1,m
      read(1,*)lc(i)
      enddo
      call branch(n,m,mf,md,mc,lc)
      stop
      end
************************************************
****    Branch and bound algorithm  ************
************************************************
      subroutine branch(n,m,mf,md,mc,cap)
      implicit Integer (a-z)
      parameter (ndim=100,mdim=100)
      integer glb,cbest,level,x(ndim)
      dimension p(ndim,mdim),lev(ndim),jlev(ndim)
      dimension ndom(ndim),nran(ndim),capo(ndim),cap(mdim)
      dimension mc(ndim,mdim),mf(ndim,ndim),md(mdim,mdim)
```

```
      logical indic
      integer*2 ihrb,ihre,iminb,imine,isecb,isece,i100b,i100e
      integer*2 year,month,day
      integer*4 bran,iter,iterb
      integer*2 std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,btse,bthu,
     -eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt
      integer*2 td,th,tm,ts,t1h
      integer*2 intm,ints,int1h,inisol,intlb
      integer*2 optd,opth,optm,opts,opt1h
      open(2,file='out2',STATUS='UNKNOWN')
      time1=0
      iter=0
      do i=1,m
       capo(i)=cap(i)
      enddo

**************************
** INITIALIZATION STEP ****
**************************
      do i=1,n
       do j=1,m
        p(i,j)=0
       enddo
      enddo
      call getdat(year,month,day)
      td=day
      std=day
      stm=month
      call gettim(ihrb,iminb,isecb,i100b)
      sths=ihrb
      stms=iminb
      stse=isecb
      sthu=i100b
      call AiBi(n,m,mf,md,capo,cbest,x)
      call gettim(ihre,imine,isece,i100e)

      iths=ihre
      itms=imine
      itse=isece
      ithu=i100e
      do i=1,m
       capo(i)=cap(i)
      enddo
      intm=imine-iminb
      ints=isece-isecb
      int1h=i100e-i100b
      inisol=cbest
      call gilmore(n,m,mf,md,mc,capo,cap,indom,ndom,nran,glb,p,
     -cbest,x,indic)
      intlb=glb
      iter=iter+1
      bran=0
      if(glb.ge.cbest)go to 4
      call choose(m,n,mf,md,p,ndom,nran,cap,ik,jk,level)
      ndom(ik)=jk
      nran(jk)=nran(jk)+1
      indom=1
```

```
      cap(jk)=cap(jk)-1
      level=1
      lev(level)=ik
      jlev(level)=jk
      bran=bran+1
**********
* STEP 1 *
**********
    1 call gilmore(n,m,mf,md,mc,capo,cap,indom,ndom,nran,glb,p,
     -cbest,x,indic)
      iter=iter+1
      if(indic)then
        iterb=iter
        call gettim(ihre,imine,isece,i100e)
      call getdat(year,month,day)
      optd=-(td-day)
      opth=ihre-ihrb
      optm=imine-iminb
      opts=isece-isecb
      opt1h=i100e-i100b

      bths=ihre
      btms=imine
      btse=isece
      bthu=i100e
      bdy=day
      bmt=month

      endif
      if(glb.ge.cbest)go to 2
      ik=0
      jk=0
      call choose(m,n,mf,md,p,ndom,nran,cap,ik,jk,level)
      ndom(ik)=jk
      nran(jk)=nran(jk)+1
      indom=indom+1
      cap(jk)=cap(jk)-1
      level=level+1
      bran=bran+1
      lev(level)=ik
      jlev(level)=jk
      if(level.eq.n .or. indic)then
        if(indic)then
        iterb=iter
        call gettim(ihre,imine,isece,i100e)
      call getdat(year,month,day)
      optd=-(td-day)
      opth=ihre-ihrb
      optm=imine-iminb
      opts=isece-isecb
      opt1h=i100e-i100b

      bths=ihre
      btms=imine
      btse=isece
      bthu=i100e
      bdy=day
```

```
       bmt=month

        else
         c=cost(lev,jlev,n,mf,md)
         if(c.lt.cbest)then
         iterb=iter
         call gettim(ihre,imine,isece,i100e)
       call getdat(year,month,day)
       optd=-(td-day)
       opth=ihre-ihrb
       optm=imine-iminb
       opts=isece-isecb
       opt1h=i100e-i100b

       bths=ihre
       btms=imine
       btse=isece
       bthu=i100e
       bdy=day
       bmt=month

          cbest=c
          do i=1,n
           x(i)=ndom(i)
          enddo
         endif
        endif
*********************
******GO TO STEP2 **
*********************
        go to 2
       else
*********************
******GO TO STEP1 **
*********************
       goto 1
       endif

**********
* STEP 2 *
**********
   2   ndom(ik)=0
       nran(jk)=nran(jk)-1
       indom=indom-1
       cap(jk)=cap(jk)+1
       p(ik,jk)=1
       ik=0
       jk=0
       level=level-1
       call choose(m,n,mf,md,p,ndom,nran,cap,ik,jk,level)
       level=level+1
       if(jk.eq.0)go to 3
       lev(level)=0
       jlev(level)=0
       level=level-1
       ndom(ik)=jk
       nran(jk)=nran(jk)+1
```

```
      indom=indom+1
      cap(jk)=cap(jk)-1
      level=level+1
      lev(level)=ik
      jlev(level)=jk
      go to 1

**********
* STEP 3 *
**********

   3  ik=lev(level)

      do j=1,m
       p(ik,j)=0
      enddo

      lev(level)=0
      jlev(level)=0
      level=level-1
      if(level.eq.0)go to 4
      ik=lev(level)
      jk=jlev(level)
      go to 2

*******
*STEP4*
*******
    4 call gettim(ihre,imine,isece,i100e)
      time=(i100e-i100b)+100*(isece-isecb+60*(imine-iminb+60
     -*(ihre-ihrb)))
      call getdat(year,month,day)
      td=-(td-day)
      eths=ihre
      etms=imine
      etse=isece
      ethu=i100e
      edy=day
      emt=month
      do while(.not.eof(2))
       read(2,*)
      enddo

      call timing(std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,btse,
     -bthu,eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt)
      th=ihre-ihrb
      tm=imine-iminb
      ts=isece-isecb
      t1h=i100e-i100b
      write(2,*)'-------------------------------------------------------'
      write(2,*)' RESULTS FOR Third GILMORE Bound Using Ai-Bi Method   '
      write(2,*)'-------------------------------------------------------'
      write(2,99)cbest,n,m
   99 format('OPTIMAL SOLUTION FOUND. IT IS :  ----->  ',i8,' N=',
     -i3,' M=',i3)
      write(*,*)'OPTIMAL SOLUTIOPN FOUND. it is',cbest
      write(2,101)iter,bran
```

```
      write(2,100)td,th,tm,ts,t1h
100   format(' TIME -->',i2,'D ',i2,'H ',i2,'Mn ',i2,'Sec ',i2,' 1/100')
101   format(' TOTAL ITERATIONS =',i10,' TOTAL BRANCHES =',i9)
      write(*,*)'Total Time =',time,' Total Number of iterations =',iter
      write(2,103)iterb,optd,opth,optm,opts,opt1h
 103  format(' OPTIMAL:',i9,'Iter',' TIME:',i2,'D ',i2'H',i2,'Mn ',
     -i2,'Sec ',i2,' 1/100')
      write(2,104)intlb,inisol,intm,ints,int1h
104   format(' STARTING LB:',i8,' SOL:',I8,' TIME:',i2,'Mn ',i2,
     -'Sec ',i2,' 1/100')
      write(*,*)'Time for starting solution =',time1
      write(*,*)'Time for optimal =',timeb,' Number of iterats =',iterb
      write(2,202)(x(i),i=1,n)
202   format(' OPTIMAL SEGENCE:',30(1x,i1))
      do i=1,n
       write(*,*)' Facility ',i,' is located at location ',x(i)
      enddo
      close(2)
      stop
      end

      subroutine timing(std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,
     -btse,bthu,eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt)
      integer*2 std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,btse,
     -bthu,eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt

      open(3,file='time.out',STATUS='UNKNOWN')
      do while(.not.eof(3))
       read(3,*)
      enddo
      write(3,*)'--------------------nb_3a-------------------------'
      write(3,1)std,stm,sths,stms,stse,sthu
  1   format(' Start  day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      write(3,2)std,stm,iths,itms,itse,ithu
  2   format(' Inital day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      write(3,3)bdy,bmt,bths,btms,btse,bthu
  3   format(' Best   day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      write(3,4)edy,emt,eths,etms,etse,ethu
  4   format(' End    day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      close(3)
      return
      end
```

Appendix B

```
      subroutine gilmore(n,m,mf,md,mc,cap,indom,ndom,nran,tcost,cbest,
     -xid,indic)
*****************************************************************
c  This subroutine computes the gilmore's bound without solving*
c  any semi-assignment problem                                  *
*****************************************************************
      implicit integer (a-z)
      parameter (ndim=100,mdim=100)
      Dimension mf(ndim,ndim),md(mdim,mdim),mc(ndim,mdim)
      dimension ndom(ndim),nran(ndim),c(ndim*(ndim-1)),d(ndim*(ndim-1))
      dimension id(ndim*(ndim-1)),cap(mdim),nd(ndim*(ndim-1))
      dimension cal1(ndim-1,ndim-1),cal2(ndim-1,ndim-1),vec(ndim-1)
      dimension dal1(ndim-1,ndim-1),dal2(ndim-1,ndim-1),xid(ndim)
      logical indic
c
c  Computing C(i,alpha) and C(alpha,i)
c
c ************************************************
      indic=.false.
      jdom=0
      do i=1,m
       if(cap(i).eq.0)then
        jdom=jdom+1
       else
        ik=i
       endif
      enddo
* to avoid continuing when there is only one usaturated location remaining
      if(m-jdom.eq.1)then
       do i=1,n
        if(ndom(i).ne.0)then
         nd(i)=ndom(i)
        else
         nd(i)=ik
        endif
       enddo
       tcost=0
       do i=1,n
        do j=1,n
         tcost=tcost+mf(i,j)*md(nd(i),nd(j))
        enddo
       enddo
       if(tcost.lt.cbest)then
        indic=.true.
        oc=cbest
        cbest=tcost
       do i=1,n
        xid(i)=nd(i)
       enddo
       endif
       return
      endif
```

```
**********************************************************************
**
      tcost=0
      if(indom.ne.0)then
       k=1
       i=1
       do while(i.le.n .and. k.le.indom)
        if(ndom(i).ne.0)then
         jj=1
         do j=1,n
          if(i.eq.j .or. ndom(j).ne.0)go to 30
          cal1(k,jj)=mf(i,j)
          cal2(k,jj)=mf(j,i)
          jj=jj+1
 30       continue
         enddo
         k=k+1
        endif
        i=i+1
       enddo
c
c  Computing D(i,alpha) and D(alpha,i)
c

       k=1
       i=1
       do while(i.le.n .and. k.le.indom)
        if(ndom(i).ne.0)then
         jj=1
         do j=1,m
          if(cap(j).ne.0)then
           do 31 tt=1,cap(j)
            dal1(jj,k)=md(ndom(i),j)
            dal2(jj,k)=md(j,ndom(i))
            jj=jj+1
 31        continue
          endif
         enddo
         k=k+1
        endif
        i=i+1
       enddo
c
c  Sorting C(i,alpha) and C(alpha,i) increasingly
c  Sorting D(i,alpha) and D(alpha,i) decreasingly
c
       do i=1,indom
        do j=1,n-indom
         vec(j)=cal1(i,j)
        enddo
        call asort(vec,n-indom)
        do j=1,n-indom
         cal1(i,j)=vec(j)
        enddo
        do j=1,n-indom
         vec(j)=cal2(i,j)
        enddo
```

```
          call asort(vec,n-indom)
          do j=1,n-indom
           cal2(i,j)=vec(j)
          enddo
          do j=1,n-indom
           vec(j)=dal1(j,i)
          enddo
          call dsort(vec,n-indom,id)
          do j=1,n-indom
           dal1(j,i)=vec(j)
          enddo
          do j=1,n-indom
           vec(j)=dal2(j,i)
          enddo
          call dsort(vec,n-indom,id)
          do j=1,n-indom
           dal2(j,i)=vec(j)
          enddo
         enddo
c
c Compute the first part of the lower bound
c
        glb=0
        do i=1,indom
         do j=1,n-indom
          glb=glb+cal1(i,j)*dal1(j,i)+cal2(i,j)*dal2(j,i)
         enddo
        enddo
       endif
c
c Construct C(alpha)- for element not in domain of alpha
c
       k=0
       do 10 i=1,n
        if(ndom(i).ne.0)goto 10
        do 11 j=1,n
         if(ndom(j).ne.0.or.i.eq.j)goto 11
         k=k+1
         c(k)=mf(i,j)
   11   continue
   10  continue
c
c Sort C(alpha) increasingly
c

       call asort(c,k)
c
c Construct D(alpha) for element not in range of alpha
c

       l=1
       do 20 i=1,m
        if(cap(i).eq.0)goto 20
        do 21 j=1,m
         if(cap(j).eq.0 .or. i.eq.j)goto 21
         d(l)=md(i,j)
         id(l)=i*(m+1)+j
```

```
         l=l+1
   21  continue
   20 continue
c
c Sort D(alpha) decreasingly
c

      call dsort(d,l-1,id)
c
c repeate the element of D(alpha) that correspond to locations
c with capacity greater than 1
c
      k=1
      do i=1,l-1
       nd(k)=d(i)
       k=k+1
       row=id(i)/(m+1)
       col=id(i)-row*(m+1)
       rep=cap(row)*cap(col)-1
       do while(rep.gt.0)
        nd(k)=d(i)
        k=k+1
        rep=rep-1
       enddo
      enddo
      do i=1,m
       rep=cap(i)**2-cap(i)
       do j=1,rep
        nd(k)=0
        k=k+1
       enddo
      enddo
c
c  Compute the second part of the lower bound
c
      do i=1,k-1
       glb=glb+c(i)*nd(i)
      enddo


c
c Add the cost of the partial assignment
      do i=1,n-1
       if(ndom(i).ne.0)then
        do j=i+1,n
         if(ndom(j).ne.0)then
          glb=glb+mf(i,j)*md(ndom(i),ndom(j))
     -           +mf(j,i)*md(ndom(j),ndom(i))
         endif
        enddo
       endif
      enddo
      tcost=glb
      return
      end
      subroutine asort(m,k)
      parameter (ndim=100)
```

```
dimension m(ndim*(ndim-1))
nt=1
do while(nt.lt.k)
  ns=nt
  nt=2*ns
  i=0
  do while (i+nt.le.k)
    call amerge(m,i+1,i+ns,i+nt)
    i=i+nt
  enddo
  if(i+ns.lt.k)call amerge(m,i+1,i+ns,k)
enddo
return
end
subroutine amerge(a,p,q,r)
implicit integer (a-z)
parameter (ndim=100)
dimension a(ndim*(ndim-1)),b(ndim*(ndim-1))
s=p
t=q+1
k=p
do while (s.le.q .and. t.le.r)
  if(a(s).le.a(t)) then
    b(k)=a(s)
    s=s+1
  else
    b(k)=a(t)
    t=t+1
  endif
  k=k+1
enddo
if(s.eq.q+1)then
  z=0
  do i=k,r
   b(i)=a(t+z)
   z=z+1
  enddo
else
  z=0
  do i=k,r
   b(i)=a(s+z)
   z=z+1
  enddo
endif
do i=p,r
 a(i)=b(i)
enddo
return
end

subroutine Dsort(m,k,n)
parameter (ndim=100)
dimension m(ndim*(ndim-1)),n(ndim*(ndim-1))
nt=1
do while(nt.lt.k)
  ns=nt
  nt=2*ns
```

```
      i=0
      do while (i+nt.le.k)
        call dmerge(m,i+1,i+ns,i+nt,n)
        i=i+nt
      enddo
      if(i+ns.lt.k)call dmerge(m,i+1,i+ns,k,n)
    enddo
    return
    end

    subroutine dmerge(a,p,q,r,c)
    implicit integer (a-z)
    parameter (ndim=100)
    dimension a(ndim*(ndim-1)),b(ndim*(ndim-1)),
   -          c(ndim*(ndim-1)),d(ndim*(ndim-1))
    s=p
    t=q+1
    k=p
    do while (s.le.q .and. t.le.r)
      if(a(s).ge.a(t)) then
        b(k)=a(s)
        d(k)=c(s)
        s=s+1
      else
        b(k)=a(t)
        d(k)=c(t)
        t=t+1
      endif
      k=k+1
    enddo
    if(s.eq.q+1)then
      z=0
      do i=k,r
       b(i)=a(t+z)
       d(i)=c(t+z)
       z=z+1
      enddo
    else
      z=0
      do i=k,r
       b(i)=a(s+z)
       d(i)=c(s+z)
       z=z+1
      enddo
    endif
    do i=p,r
     a(i)=b(i)
     c(i)=d(i)
    enddo
    return
    end
```

Appendix C

```
      subroutine gilmore(nn,m,mf,md,mc,cap,indom,ndom,nran,tcost,ip,
     -cbest,xid,indic)
************************************************************
c  This subroutine computes the gilmore's bound by solving one *
c  semi-assignment problem                                     *
************************************************************
      implicit integer (a-z)
      parameter (ndim=100,mdim=100)
      Dimension mf(ndim,ndim),md(mdim,mdim),mc(ndim,mdim),lc(mdim)
      dimension ndom(ndim),nran(ndim),ca(ndim*(ndim-1)),d(ndim*(ndim-1))
      dimension id(ndim*(ndim-1)),cap(mdim),nd(ndim*(ndim-1)),xid(ndim)
      dimension cal1(ndim,ndim),cal2(ndim,ndim),vec(ndim)
      dimension dal1(ndim,ndim),dal2(ndim,ndim)
      dimension z(ndim,ndim),z1(ndim,ndim),z2(ndim,ndim),ip(ndim,mdim)
      logical indic
c ************************************************
c refer to the relax code for the following:       *
c ************************************************
      PARAMETER (MAXNN=10000, MAXNA=70000)
      INTEGER STARTN(MAXNA),ENDN(MAXNA),C(MAXNA)
      COMMON /INPUT/N,NA,LARGE
      COMMON /ARRAYS/STARTN/ARRAYE/ENDN/ARRAYC/C
      INTEGER U(MAXNA),B(MAXNN)
      COMMON /ARRAYU/U/ARRAYB/B
      INTEGER X(MAXNA),RC(MAXNA)
      COMMON /ARRAYX/X/ARRAYRC/RC
c ************************************************
      indic=.false.
      jdom=0
      do i=1,m
       if(cap(i).eq.0)then
        jdom=jdom+1
       else
        ik=i
       endif
      enddo
c to avoid continuing when there is only one usaturated location remaining
      if(m-jdom.eq.1)then
       do i=1,nn
        if(ndom(i).ne.0)then
         nd(i)=ndom(i)
        else
         nd(i)=ik
        endif
       enddo
       tcost=0
       do i=1,nn
        do j=1,nn
         tcost=tcost+mf(i,j)*md(nd(i),nd(j))
        enddo
       enddo
       if(tcost.lt.cbest)then
        indic=.true.
```

```
        oc=cbest
        cbest=tcost
        do i=1,nn
         xid(i)=nd(i)
        enddo
       endif
       return
      endif

      do i=1,nn
       do j=1,m
        z(i,j)=0
       enddo
      enddo

      do i=1,nn
       do j=1,nn
        cal1(i,j)=0
        cal2(i,j)=0
       enddo
      enddo

      do i=1,m
       do j=1,m
        dal1(i,j)=0
        dal2(i,j)=0
       enddo
      enddo

      n=nn
      ii=1
      do i=1,n
       k=1
       if(ndom(i).eq.0)then
        do j=1,n
         if(i.eq.j .or. ndom(j).ne.0)go to 30
          cal1(ii,k)=mf(i,j)
          cal2(ii,k)=mf(j,i)
          k=k+1
30       continue
        enddo
        ii=ii+1
       endif
      enddo

      ii=1
      do i=1,m
       if(cap(i).ne.0)then
        jj=1
        do j=1,m
         rep=cap(j)
         if(i.eq.j)rep=rep-1
         do tt=1,rep
          dal1(jj,ii)=md(i,j)
          dal2(jj,ii)=md(j,i)
          jj=jj+1
31        continue
```

```
          enddo
         enddo
         ii=ii+1
        endif
       enddo

       do i=1,n-indom
        do j=1,n-1-indom
         vec(j)=cal1(i,j)
        enddo
        call asort(vec,n-1-indom)
        do j=1,n-1-indom
         cal1(i,j)=vec(j)
        enddo
        do j=1,n-1-indom
         vec(j)=cal2(i,j)
        enddo
        call asort(vec,n-1-indom)
        do j=1,n-1-indom
         cal2(i,j)=vec(j)
        enddo
        do j=1,n-1-indom
         vec(j)=dal1(j,i)
        enddo
        call dsort(vec,n-1-indom,id)
        do j=1,n-1-indom
         dal1(j,i)=vec(j)
        enddo
        do j=1,n-1-indom
         vec(j)=dal2(j,i)
        enddo
        call dsort(vec,n-1-indom,id)
        do j=1,n-1-indom
         dal2(j,i)=vec(j)
        enddo
       enddo

       do i=1,n-indom
        do j=1,m-jdom
         z1(i,j)=0
         z2(i,j)=0
         do k=1,n-1
          z1(i,j)=z1(i,j)+cal1(i,k)*dal1(k,j)
          z2(i,j)=z2(i,j)+cal2(i,k)*dal2(k,j)
         enddo
********   here we consider that tao is equal to one half
*****************
         z(i,j)= (z1(i,j)+z2(i,j))/2
        enddo
       enddo

c here we consider the interaction between assigned an nonassigned
facilities
       k=0
       do i=1,n
        if(ndom(i).eq.0)then
         k=k+1
```

```
          l=0
          do j=1,m
           if(cap(j).ne.0)then
            l=l+1
            do t=1,n
             if(ndom(t).ne.0)then
              z(k,l)=z(k,l)+mf(i,t)*md(j,ndom(t))+mf(t,i)*md(ndom(t),j)
             endif
            enddo
           endif
          enddo
         endif
        enddo
        k=0
c Setting the problem for the relax routine
        na=(n-indom)*(m-jdom)
        k=1
        do i=1,n-indom
         do j=1,m-jdom
          startn(k)=i
          endn(k)=n-indom+j
          c(k)=z(i,j)
          u(k)=1
          k=k+1
         enddo
         b(i)=-1
        enddo
        jj=1
        do j=1,m
         if(cap(j).ne.0)then
          b(n-indom+jj)=cap(j)
          jj=jj+1
         endif
        enddo
        n=nn+m-indom-jdom
        call assign(tcost)
        do i=1,nn-1
         if(ndom(i).ne.0)then
          do j=i+1,nn
           if(ndom(j).ne.0 .and. j.ne.i)then
            tcost=tcost+mf(i,j)*md(ndom(i),ndom(j))
     -                 +mf(j,i)*md(ndom(j),ndom(i))
           endif
          enddo
         endif
        enddo
        return
        end
```

Appendix D

```
      subroutine gilmore(nn,m,mf,md,mc,cap,ocap,indom,ndom,nran,tcost,p,
     -cbest,xid,indic)
************************************************************
c  This subroutine computes the gilmore's bound by solving n^2 *
c  semi-assignment problems                                    *
************************************************************
      implicit integer (a-z)
      parameter (ndim=100,mdim=100)
      Dimension mf(ndim,ndim),md(mdim,mdim),mc(ndim,mdim),lc(mdim)
      dimension ndom(ndim),nran(ndim),ca(ndim*(ndim-1)),d(ndim*(ndim-1))
      dimension xid(ndim),ocap(mdim),cap(mdim),xnd(ndim)
      dimension cal1(ndim-1,ndim-1),cal2(ndim-1,ndim-1),vec(ndim-1)
      dimension dal1(ndim-1,ndim-1),dal2(ndim-1,ndim-1),p(ndim,mdim)
      dimension z(ndim,ndim),z1(ndim,ndim),z2(ndim,ndim)
      logical indic

      PARAMETER (MAXNN=10000, MAXNA=70000)
      INTEGER STARTN(MAXNA),ENDN(MAXNA),C(MAXNA)
      COMMON /INPUT/N,NA,LARGE
      COMMON /ARRAYS/STARTN/ARRAYE/ENDN/ARRAYC/C
      INTEGER U(MAXNA),B(MAXNN)
      COMMON /ARRAYU/U/ARRAYB/B
c ************************************************
      indic=.false.
      jdom=0
      do i=1,m
       if(ocap(i).eq.0)then
        jdom=jdom+1
       else
        ik=i
       endif
      enddo
* to avoid continuing when there is only one usaturated location remaining
      if(m-jdom.eq.1)then
       do i=1,nn
        if(ndom(i).ne.0)then
         xnd(i)=ndom(i)
        else
         xnd(i)=ik
        endif
       enddo
       tcost=0
       do i=1,nn
        do j=1,nn
         tcost=tcost+mf(i,j)*md(xnd(i),xnd(j))
        enddo
       enddo
       if(tcost.lt.cbest)then
        indic=.true.
        cbest=tcost
        do i=1,nn
         xid(i)=xnd(i)
        enddo
       endif
```

```
        return
       endif
**********************************************************************
**
      n=nn+m
      do i=1,nn
       do j=1,m
c
c  Obtain the costs for the semi-assignment problem relative to Xij
c
        do k=1,nn
         do l=1,m
          if(i.eq.k .or. ndom(k).ne.0)then
           z1(k,l)=10000
           if((i.eq.k .and. j.eq.l).or.ndom(k).eq.l)z1(k,l)=0
          else
           z1(k,l)=mf(i,k)*md(j,l)
          endif
         enddo
        enddo
        do k=1,nn
         do l=1,m
         enddo
        enddo
        na=nn*m
        kk=1
c
c  Set the problem for solution with relax4
c
        do k=1,nn
         do l=1,m
          startn(kk)=k
          endn(kk)=nn+l
          c(kk)=z1(k,l)
          u(kk)=1
          kk=kk+1
         enddo
         b(k)=-1
        enddo
        do l=1,m
         b(nn+l)=cap(l)
        enddo
c
c If facility i has already been assigned then force it to its location by
c forbiding the other possibilities.
c
        if(j.ne.ndom(i))then
         call assign(tcost)
         z(i,j)=tcost
        endif
       enddo
      enddo
c
c Set the data for the last semi-assignment problem
c
      do i=1,nn
       if(ndom(i).ne.0)then
```

```
   do j=1,m
    z(i,j)=10000
    if(ndom(i).eq.j)z(i,j)=0
   enddo
  endif
enddo
do i=1,nn
 if(ndom(i).eq.0)then
  do j=1,m
   if(cap(j).ne.0)then
    do k=1,nn
     if(ndom(k).ne.0)then
      z(i,j)=z(i,j)+mf(i,k)*md(j,ndom(k))+mf(k,i)*md(ndom(k),j)
     endif
    enddo
   endif
  enddo
 endif
enddo

do i=1,nn
 if(ndom(i).ne.0)then
  do j=1,m
   z(i,j)=10000
   if(ndom(i).eq.j)z(i,j)=0
  enddo
 endif
enddo

na=nn*m
kk=1
do k=1,nn
 do l=1,m
  startn(kk)=k
  endn(kk)=nn+l
  c(kk)=z(k,l)
  u(kk)=1
  kk=kk+1
 enddo
 b(k)=-1
enddo
do l=1,m
 b(nn+l)=cap(l)
enddo
call assign(tcost)
do i=1,n
 if(ndom(i).ne.0)then
  do j=1,n
   if(ndom(j).ne.0 .and. j.ne.i)then
    tcost=tcost+mf(i,j)*md(ndom(i),ndom(j))
   endif
  enddo
 endif
enddo
return
end
```

```
      integer function cost(m,t,n,f,d)
      parameter (ndim=100,mdim=100)
      integer f(ndim,ndim),d(mdim,mdim),m(ndim),t(ndim)
      integer sol
      sol=0
      do i=1,n-1
       do j=i+1,n
      sol=sol+f(m(i),m(j))*d(t(i),t(j))
      sol=sol+f(m(j),m(i))*d(t(j),t(i))
       enddo
      enddo
      cost=sol
      return
      end

      subroutine choose(m,n,mf,md,p,ndom,nran,cap,i0,j0,level)
      implicit integer (a-z)
      parameter (ndim=100,mdim=100)
      dimension mf(ndim,ndim),md(mdim,mdim),p(ndim,mdim)
      dimension ndom(ndim),nran(mdim),cap(mdim)

      if(level.eq.0)then
1     maxmin=0
      do i=1,n
      sum=0
      do j=1,n
       sum=sum+mf(i,j)
      enddo
      if(maxmin.lt.sum)then
       maxmin=sum
       i0=i
      endif
      enddo
      maxmin=10000000
      do j=1,m
       if(p(i0,j).eq.0)then
        sum=0
        do i=1,m
         sum=sum+md(j,i)
        enddo
        if(maxmin.gt.sum)then
         maxmin=sum
         j0=j
        endif
       endif
      enddo
      if(j0.eq.0)then
       i0=-1
       return
      endif
     else
      maxmin=0
      do i=1,n
       if(ndom(i).eq.0)then
        sum=0
        do j=1,n
         if(ndom(j).ne.0)sum=sum+mf(i,j)
```

```
    enddo
    if(maxmin.lt.sum)then
     maxmin=sum
     i0=i
    endif
   endif
  enddo

  maxmin=1000000
  do j=1,m
   if(cap(j).ne.0 .and. p(i0,j).eq.0)then
    sum=0
    do i=1,m
     if(nran(i).ne.0)sum=sum+md(j,i)
    enddo
    if(maxmin.gt.sum)then
     maxmin=sum
     j0=j
    endif
   endif
  enddo
endif
return
end

subroutine free(m,jk,mat,mf,md,cap,val,k)
parameter (ndim=100,mdim=100)
Dimension mf(ndim,ndim),md(mdim,mdim),mat(ndim,ndim)
integer sol,cap(mdim),val,comp

val=1000000
do j=1,m
 if(cap(j).gt.0)then
  mat(k+1,2)=j
  sol=comp(mat,k+1,mf,md)
  if(sol.lt.val)then
   val=sol
   jk=j
  endif
 endif
enddo
return
end

function comp(m,n,f,d)
parameter (ndim=100,mdim=100)
integer f(ndim,ndim),d(mdim,mdim),m(ndim,ndim)
integer comp,sol
sol=0
do i=1,n-1
 do j=i+1,n
  sol=sol+f(m(i,1),m(j,1))*d(m(i,2),m(j,2))
  sol=sol+f(m(j,1),m(i,1))*d(m(j,2),m(i,2))
 enddo
enddo
comp=sol
return
```

```
end

subroutine order(ord,i)
parameter (ndim=100)
integer ord(ndim)
ind=ord(i)
ord(i)=ord(i+1)
ord(i+1)=ind
return
end
```

Appendix E

```
      subroutine AiBi(n,m,mf,md,cap,cost,ndom)
      implicit integer (a-z)
*****************************************************************
c  This subroutine computes a starting solution using Ai-Bj Rule*
*****************************************************************
      parameter (ndim=100,mdim=100)
      Dimension mf(ndim,ndim),md(mdim,mdim),cap(mdim)
      dimension ndom(ndim),nran(ndim),A(ndim*(ndim-1)),B(ndim*(ndim-1))

      indom=0
      inran=0
      do while(indom.lt.n)
c  Compute Ai
      k=1
      Do i=1,n
      a(i)=10000
      if(ndom(i).eq.0)then
      a(i)=0
       Do p=1,n
        if(ndom(p).eq.0)then
         a(i)=a(i)+mf(i,p)+mf(p,i)
        endif
       enddo
      endif
      enddo

c Compute Bj
      do j=1,m
       b(j)=-1
       if(cap(j).ne.0)then
        b(j)=0
        do q=1,m
         if(nran(q).eq.0 .and. cap(q).ne.0)then
          b(j)=b(j)+md(j,q)+md(q,j)
         endif
        enddo
       endif
      enddo

c Find min Ai

      imin=1
      do i=1,n
       if(a(imin).gt.a(i))imin=i
      enddo

c Find max Bj

      jmax=1
      do j=1,m
       if(b(jmax).lt.b(j))jmax=j
      enddo

c Assign facility imin to location jmax
```

```
      ndom(imin)=jmax
      nran(jmax)=imin
      cap(jmax)=cap(jmax)-1
      indom=indom+1
      if(cap(jmax).eq.0)inran=inran+1
      enddo
      cost=0
      do i=1,n-1
       do j=i+1,n
        cost=cost+mf(i,j)*md(ndom(i),ndom(j))
     -           +mf(j,i)*md(ndom(j),ndom(i))
       enddo
      enddo
      return
      end
```

Appendix F

```
      subroutine incdeg(m,n,mf,md,cap,sol,x)
*****************************************************************
c  This subroutine computes a starting solution using the       *
c  method of Increasing Degrees of Freedom                      *
*****************************************************************
      parameter (ndim=100,mdim=100)
      Dimension mf(ndim,ndim),md(mdim,mdim),mat(ndim,ndim)
      integer sol,cap(mdim),comp,val,ord(ndim),capo(mdim),x(ndim)
      logical change
      do i=1,m
       capo(i)=cap(i)
      enddo
      do i=1,n
       ord(i)=i
      enddo
      minim=1000000
      nn=n/2
      do it=1,n
       do i=1,m
        cap(i)=capo(i)
       enddo

       k=0
       do i=1,n
        change=.false.
        ik=ord(i)
        mat(k+1,1)=ik
        call free(m,jk,mat,mf,md,cap,val,k)
        mat(k+1,2)=jk
        min=val
         do l=1,k
          lo=mat(l,1)
          mat(l,1)=ik
          mat(k+1,1)=lo
          call free(m,jo,mat,mf,md,cap,val,k)
          if(val.lt.min)then
           min=val
           nj=jo
           ol=l
           change=.true.
          endif
          mat(l,1)=lo
          mat(k+1,1)=ik
         enddo
       if(change)then
        mat(k+1,2)=mat(ol,2)
        mat(ol,2)=nj
        cap(nj)=cap(nj)-1
       else
        mat(k+1,2)=jk
        cap(jk)=cap(jk)-1
       endif
       k=k+1
       enddo
```

```
 sol=comp(mat,n,mf,md)
 call order(ord,it)
 if(sol.lt.minim)then
  do i=1,n
   x(mat(i,1))=mat(i,2)
  enddo
  minim=sol
 endif
enddo
sol=minim
return
end
```

Appendix G

```
********************** Initialize LINDO
*********************************
      subroutine initial()
      include 'lindps~1.h'
      character*8 KFNAME
      KFNAME='DATA.MPS'
      call ILINDO()
      call INIT()
      call lunopn(1,8,LOC(KFname),1,0,5,6)
      call rdmps(1,1)
      call quiet(-1)
      return
      end
*************************************************************
      subroutine branch(n,m,mf,md,mc,cap)
      implicit integer (a-z)
      include 'lindps~1.h'
      parameter (ndim=100,mdim=100)
      Dimension mf(ndim,ndim),md(mdim,mdim),mc(ndim,mdim),cap(mdim)
      dimension br(ndim*mdim),x(ndim)
      integer*4 stack(ndim*mdim),bnd(ndim*mdim),bound
      integer*4 imax,lb
      real*4 sol(1000)
      logical inte

      integer*2 ihrb,ihre,iminb,imine,isecb,isece,i100b,i100e
      integer*2 year,month,day
      integer*4 bran,iter,iterb
      integer*2 std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,btse,bthu,
     -eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt
      integer*2 td,th,tm,ts,t1h
      integer*2 intm,ints,int1h,inisol,intlb
      integer*2 optd,opth,optm,opts,opt1h
      open(2,file='out2',STATUS='UNKNOWN')
      iter=0
      time1=0

      call readsol(m,n,sol,imax,bound,inte)

      intlb=bound
      bran=0
      iter=iter+1

      if(.not.inte)then
       write(*,*)'solution is initially optimal  ',bound
       return
      else

       call getdat(year,month,day)
       td=day
       std=day
       stm=month
       call gettim(ihrb,iminb,isecb,i100b)
       sths=ihrb
```

```
        stms=iminb
        stse=isecb
        sthu=i100b
        call incdeg(m,n,mf,md,cap,cbest,x)
        call gettim(ihre,imine,isece,i100e)
        iths=ihre
        itms=imine
        itse=isece
        ithu=i100e
        intm=imine-iminb
        ints=isece-isecb
        int1h=i100e-i100b
        inisol=cbest

        level=0
        lb=bound
********
* Step 1
********
   1    level=level+1
        stack(level)=imax
        bnd(level)=bound
        call setslb(imax,1.)
        br(level)=1
        bran=bran+1
   4    call readsol(m,n,sol,imax,bound,inte)
        iter=iter+1
        if(.not.inte)then
         if(bound.lt.cbest)then
          cbest=bound
          iter=iter+1
          iterb=iter
          call gettim(ihre,imine,isece,i100e)
          call getdat(year,month,day)
          optd=-(td-day)
          opth=ihre-ihrb
          optm=imine-iminb
          opts=isece-isecb
          opt1h=i100e-i100b
          bths=ihre
          btms=imine
          btse=isece
          bthu=i100e
          bdy=day
          bmt=month
         endif
         if(br(level).eq.1)then
          goto 2
         else
          goto 3
         endif
        else
         lb=bound
         if(lb.lt.cbest)then
          goto 1
         else
          if(br(level).eq.1)then
```

```
          goto 2
         else
          goto 3
         endif
        endif
       endif
********
* STEP 2
********
   2    imax=stack(level)
        call setslb(imax,0.)
        call setsub(imax,0.)
        br(level)=0
        goto 4
********
* STEP 3
********
   3    imax=stack(level)
        call setsub(imax,1.)
        level=level-1
        if(level.eq.0)goto 5
        lb=bnd(level)
        if(br(level).eq.1)then
         goto 2
        else
         goto 3
        endif
       endif
   5   write(*,*)'OPtimal solution found  it is ',cbest
*******
*STEP4*
*******
       call gettim(ihre,imine,isece,i100e)
       time=(i100e-i100b)+100*(isece-isecb+60*(imine-iminb+60
      -*(ihre-ihrb)))
       call getdat(year,month,day)
       td=-(td-day)
       eths=ihre
       etms=imine
       etse=isece
       ethu=i100e
       edy=day
       emt=month
       th=ihre-ihrb
       tm=imine-iminb
       ts=isece-isecb
       t1h=i100e-i100b
       do while(.not.eof(2))
        read(2,*)
       enddo

       call timing(std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,btse,
      -bthu,eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt)
       write(2,*)'-------------------------------------------------'
       write(2,*)' RESULTS FOR RLT Using INC DEG of FREEDOM          '
       write(2,*)'-------------------------------------------------'
       write(2,99)cbest,n,m
```

```
   99 format('OPTIMAL SOLUTION FOUND. IT IS :  ----->  ',i8,' N=',
     -i3,' M=',i3)
      write(*,*)'OPTIMAL SOLUTIOPN FOUND. it is',cbest
      write(2,101)iter,bran
      write(2,100)td,th,tm,ts,t1h
  100  format(' TIME -->',i2,'D ',i2,'H ',i2,'Mn ',i2,'Sec ',i2,' 1/100')
  101  format(' TOTAL ITERATIONS =',i10,' TOTAL BRANCHES =',i9)
      write(*,*)'Total Time =',time,' Total Number of iterations =',iter
      write(2,103)iterb,optd,opth,optm,opts,opt1h
  103 format(' OPTIMAL:',i9,'Iter',' TIME:',i2,'D ',i2'H',i2,'Mn ',
     -i2,'Sec ',i2,' 1/100')
      write(2,104)intlb,inisol,intm,ints,int1h
  104  format(' STARTING LB:',i8,' SOL:',I8,' TIME:',i2,'Mn ',i2,
     -'Sec ',i2,' 1/100')
      write(*,*)'Time for starting solution =',time1
      write(*,*)'Time for optimal =',timeb,' Number of iterats =',iterb
      write(2,202)(x(i),i=1,n)
  202  format(' OPTIMAL SEGENCE:',30(1x,i1))
      do i=1,n
       write(*,*)' Facility ',i,' is located at location ',x(i)
      enddo
      close(2)
      return
      end

      subroutine timing(std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,
     -btse,bthu,eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt)
      integer*2 std,stm,bdy,bmt,sths,stms,stse,sthu,bths,btms,btse,
     -bthu,eths,etms,etse,ethu,iths,itms,itse,ithu,edy,emt
      open(3,file='time.out',STATUS='UNKNOWN')

      do while(.not.eof(3))
       read(3,*)
      enddo
      write(3,*)'-------------------- RLT -------------------------'
      write(3,1)std,stm,sths,stms,stse,sthu
    1  format(' Start  day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      write(3,2)std,stm,iths,itms,itse,ithu
    2  format(' Inital day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      write(3,3)bdy,bmt,bths,btms,btse,bthu
    3  format(' Best   day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      write(3,4)edy,emt,eths,etms,etse,ethu
    4  format(' End    day: ',i2,'/',i4,'   time: ',i2,' H ',i2,' Mn ',
     -i2,' Sec ',i2)
      close(3)
      return
      end

*****************************************************************
*****************************************************************
      Subroutine readsol(m,n,sol,imax,bound,inte)
      include 'lindps-1.h'
      integer debut,fin
      logical inte
```

```
real*4 val,dual,max,sol(1000),eps,redc(1000),p(100,10),c(100,10)
integer*4 bound,imax
eps=1e-14
call go(200000000,imax)
if(imax.eq.4)then
 imax=0
 max=0
 inte=.false.
 call reprow(1,val,dual)
 bound=int(val)
 if(val-bound.gt.0)bound=bound+1
 debut=1
 fin=m*n
 do i=debut,fin
  call repvar(i,val,dual)
  sol(i)=val
  redc(i)=dual
  if(val.lt.1-eps .and. val.gt.0+eps) then
   inte=.true.
   if(val.gt.max) then
    max=val
    imax=i
   endif
  endif
 enddo
 do k=debut,fin
  i=k/m
  j=k-i*m
  if(j.ne.0)then
   i=i+1
  else
   j=m
  endif
  c(i,j)=redc(k)
 enddo
 do i=1,n
  do j=1,m
   min1=1000000
   min2=1000000
   do k1=1,m
    if(k1.ne.j .and. min1.gt.c(i,k1))min1=c(i,k1)
   enddo
   do k2=1,n
    if(k2.ne.i .and. min2.gt.c(k2,j))min2=c(k2,j)
   enddo
   p(i,j)=min1+min2
  enddo
 enddo
 max=-10000000
 do i=1,n
  do j=1,m
   if(p(i,j).gt.max)then
    max=p(i,j)
    i0=i
    j0=j
   endif
  enddo
```

```
        enddo
        imax=(i0-1)*m+j0
      else
        if(imax.eq.2)then
          bound=100000
          return
        endif
        write(*,*)'Linear Solution Not optimal'
        stop
      endif
      return
      end

**********************************************
```

REFERENCES

1 Adams W. and Johnson T. "Improved linear programming based lower bounds for the quadratic assignment problem", in *Quadratic Assignment and Related Problems, P. Pardalos and H. Wolkowicz eds. DIMACS series in Discrete Mathematics and Theoretical Computer Science*, **16**(1994) 43-77 *American Mathematical Society*

2 Ahuja R., Orlin J. and Tivari A. "A greedy genetic algorithm for the quadratic assignment problem", Report 3826-95 Sloan School of Management MIT 1995

3 Armour G. and Buffa E. "A heuristic and simulative approach to relative location of facilities", *Management Science* **9**(1963) 294-309

4 Assad A. and Xu W. "On lower bounds for a class of quadratic 0-1 programs", *Operations Research Letters* **4**(1985) 175-180

5 Balas E. and Mazzola J. "Quadratic 0-1 programming by a new linearization", presented at the joint *ORSA/TIMS* National meeting 1980 Washington D.C.

6 Balinski M. and Russakoff A. "On the assignment polytope", *SIAM Review* **16**(1974) $n°4$ 516-525

7 Ball M., Kaku B. and Vakhutinsky A. "Network-based formulation of the quadratic assignment problem", *European Journal of Operational Research* **104**(1994) 241-249

8 Battiti R. and Tecchiolli G. " The reactive tabu search", *ORSA Journal on Computing* **6**(1994) 126-140

9 Bazaraa M. "The quadratic set covering (assignment) problem: Applications and computation", Report presented to the *National Science Foundation under grant* GK-38337 1975

10 Bazaraa M. and Elshafei A. "Exact and heuristic procedures for the quadratic assignment problem", *Naval Research Logistics Quarterly* **26**(1979) 109-121

11 Bazaraa M. and Kirca O. "Branch and bound heuristics for solving the quadratic assignment problem", *Naval Research Logistics Quarterly* **30**(1983) 287-304

12 Bazaraa M. and Sherali H. "Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem", *Naval Research Logistics Quarterly* **27**(1980) 29-41

13 Bazaraa M. and Sherali H. "On the use of exact and heuristic cutting plane methods for the quadratic assignment problem", *Journal of the operational research society* **33**(1982) 991-1003

14 Breuer M. "The formulation if some allocation and connection problems as integer programs", *Naval Research Logistics Quarterly* **13**(1966) 83-95

15 Brown D., Huntley C. and Spillane A. "A parallel genetic heuristic for the quadratic assignment problem", in: *Proceedings of the* 3^{rd} *conference on Genetic Algorithms*, 406-415, Arlington 1989

16 Bruijis P. "On the quality of heuristic solutions to a 19×19 quadratic assignment problem", *European Journal of Operational Research* **17**(1984) 21-30

17 Brungger A., Marzetta A., Clausen J. and Perregaard M. "Joining forces in solving large-scale quadratic assignment problems in parallel", *Proceedings of the* 11^{th} *International Parallel Processing Symposium IEEE* (1997) 418-427

18 Burkard R. "Die storungsmethode zur losunng quadratisher Zuordnungsprobleme", *Operations Research Verfahren* **16**(1973) 84-108 in German

19 Burkard R. "Quadratic assignment problem", *European Journal of Operational Research* **15**(1984) 283-289

20 Burkard R. "Locations with spatial interactions: The quadratic assignment problem", in *Discrete Location Theory* P. Mirchandani and R. Francis eds. Wiley 1991, 387-437

21 Burkard R. and Bonniger T. "A heuristic for quadratic boolean programs with applications to quadratic assignment problems", *European Journal of Operational Research* **13**(1983) 374-386

22 Burkard R., Cela E., Rote G. and Woeginger G. "The quadratic assignment problem with a monotone anti-Monge and a symmetric Toeplitz matrix: Easy and had cases", *Mathematical Programming* **82**(1998) 125-158

23 Burkard R. and Derigs U. "*Assignment and matching problems: Solution methods with Fortran-programs*", Lecture Notes in Economics and Mathematical Systems 184 Springer-Verlag, Berlin 1980

24 Burkard R., Karisch S. and Rendl F. "QAPLIB-A quadratic assignment problem library",*Journal of Global Optimization* **10**(1997) 391-403. Available on line at URL: http://opt.math.tu.graz.ac.at/~karisch/qaplib/

25 Burkard R. and Rendl F. "A thermodynamically motivated simulated procedure for combinatorial optimization problems", *European Journal of Operational Research* **17**(1983) 169-174

26 Burkard R. and Stratmann K. "Numerical investigation on quadratic assignment problems", *Naval Research Logistics Quarterly* **25**(1978) 129-148

27 Burkov V., Rubinstein M., and Sokolov V. " Some problems in optimal allocation of large volume memories", *Avtomatika i Telemekhanika* **9**(1969) 83-91, in Russian

28 Carraresi P. and Malucelli F. "A new lower bound for the quadratic assignment problem", *Operations Research* **40**(1992) suppl. $n°1$ *S22-S27*

29 Carraresi P. and Malucelli F. "A reformulation scheme and new lower bounds for the QAP", in *Quadratic Assignment and Related Problems, P. Pardalos and H. Wolkowicz eds. DIMACS series in Discrete Mathematics and Theoretical Computer Science*, **16**(1994) 147-160 *American Mathematical Society*

30 Chakrapani J., and Skorin-Kapov J. "Massively parallel tabu search for the quadratic assignment problem", *Technical Report HAR*-91-06, Harriman School for Management and Policy,New York 1991.

31 Chakrapani J. and Skorin-Kapov J. "A connectionist approach to the quadratic assignment problem", *Computers and Operations Research* **19**(1992) 287-295

32 Chakrapani J. and Skorin-Kapov J. "A constructive method for improving lower bounds for a class of quadratic assignment problems", *Operations Research* **42**(1994) $n°5$ 837-845

33 Chan A. and Francis R. "A least total distance facility configuration problem involving lattice points", *Management Science* **22**(1976) 778-787

34 Chen B. "Special cases of the quadratic assignment problem", *European Journal of Operational Research* **81**(1995) 410-419

35 Christofides N. and Benavent E. "An exact algorithm for the quadratic assignment problem on a tree", *Operations Research* **37**(1989) $n°5$ 760-768

36 Christofides N. and Gerrard M. "Special cases of the quadratic assignment problem", *Management Science Research Report* 391, Carnegie-Mellon University, Pittsburgh, PA, 1976

37 Clausen J. and Perregaad M. "Solving large quadratic assignment problems in parallel" **to appear in** *Computational Optimization and Applications*

38 Conrad K. "*Das quadratishe zuordnungsproblem und zwei seiner spezialfalle*", Mohr Siebeck Publishers, Tubingen Germany 1971 in German

39 Conway R. and Maxwell W. " A note on the assignment of facility location", *Journal of Industrial Engineering* **12**(1961) 34-36

40 Crouse J. and Pardalos P. "A parallel algorithm for the quadratic assignment problem", in: *Proceedings of supercomputing* 1989, 351-360, *ACM*

41 Cung V., Mautor T., Michelon P. and Tavares A. "A scatter search based approach for the quadratic assignment problem", *Proceedings of the IEEE international conference on evolutionary computation* (1997) 165-169

42 Cyganski D., Vaz R. and Virball V. "Quadratic assignment problems generated with Palubetskis algorithm are degenerate", *IEEE Transactions on Circuits and Systems-I Fundamental theory and applications* **41**(1994) 481-484

43 Dickey J. and Hopkins J. "Campus building arrangement using TOPAZ", *Transportation Research* **6**(1972) 59-68

44 Drezner Z. "Lower bounds based on linear programming for the quadratic assignment problem", *Computational Optimization and Applications* **4**(1995) 159-165

45 Edwards C. "A branch and bound algorithm for the Koopmans and Beckmann quadratic assignment problem", *Mathematical Programming Study* **13**(1980) 35-52

46 Edwards H., Gillett B. and Hale M. "Modular allocation technique (MAT)", *Management Science* **17**(1970) 161-169

47 Elshafei A. "Hospital Layout as a quadratic assignment problem", *Operational Research Quarterly* **28**(1977) $n°1$ *ii* 167-179

48 Fathi Y. and Ginjupalli K. "A mathematical model and heuristic procedure for the turbine balancing problem", *European Journal of Operational Research* **63**(1993) 336-342

49 Finke G., Burkard R. and Rendl F. "Quadratic assignment problem", *Annals of discrete mathematics* **31**(1987) 61-82

50 Fleurent C. and Ferland J. "Genetic Hybrids for the quadratic assignment problem", in *Quadratic Assignment and Related Problems, P. Pardalos and H. Wolkowicz eds. DIMACS series in Discrete Mathematics and Theoretical Computer Science*, **16**(1994) 173-187 *American Mathematical Society*

51 Frieze A. and Yedegar J. "On the quadratic assignment problem", *Discrete Applied Mathematics* **5**(1983) 89-98

52 Francis R. and White J. "*Facility Layout and Location: An analytical approach*", Printice-Hall 1974

53 Fu M. and Kaku B. " Minimizing work-in-process and material handling in the facilities layout problem", *IEE Transactions* **29**(1977) 29-36

54 Gavett J. and Plyter N. "The optimal assignment of facilities to locations by branch and bound", *Operations Research* **14**(1966) 210-232

55 Geoffrion A. and Graves G. "Scheduling parallel production lines with changeover costs: Practical application of quadratic assignment/ LP approach", *Operations Research* **24**(1976) 595-610

56 Gilmore P.C. "Optimal and suboptimal algorithms for the quadratic assignment problem", *SIAM Journal of Applied Mathematics* **10**(1962) 305-313

57 Goemans M. and Williamson D. "Improved approximation algorithms for maximum cut and satifiability problems using semi-definite programming", *Journal of the ACM* **42**(1995) 1115-1145

58 Graves G. and Whinston A. "An algorithm for the quadratic assignment problem", *Management Science* **17**(1970) 453-471

59 Haghani A. and Chen M. " Optimizing gate assignments at airport terminals", *Transportation Research* **32**(1998) $n°6$ 437-454

60 Hadley S., Rendl F. and Wolkowicz H. "Bounds for the quadratic assignment problem using continuous optimization techniques", in *Proceedings of the* 1^{st} *International Integer programming and Combinatorial Optimization Conference* (*IPCO*), 1990 237-248

61 Hadley S., Rendl F. and Wolkowicz H. "A new lower bound via projection for the quadratic assignment problem", *Mathematics of Operations Research* **17**(1992) $n°3$ 727-739

62 Hahn P. and Grant T. "Lower bounds for the quadratic assignment problem based upon a dual formulation", *Operations Research* **46**(1998) $n°6$ 912-922

63 Hahn M., Grant T. and Hall N. "Solution of the quadratic assignment problem using the Hungarian method", **to appear in** *European Journal of Operational Research*

64 Hanan M. and Kurtzberg J. "A review of the placement and quadratic assignment problems", *SIAM Review* **14**(1972) 324-342

65 Heider C. "A decomposition procedure for the quadratic assignment problem", Professional paper $n°100$ (1972)

66 Heider C. "A computationally simplified pair exchange algorithm for the quadratic assignment problem", Professional paper $n°101$ (1972)

67 Heider C. "An n-step variable search algorithm for component placement", *Naval Research Logistics Quarterly* **20**(1973) 699-724

68 Hillier F. "Quantitative tools for plant layout analysis", *Journal of Industrial Engineering* **14**(1966) 33-40

69 Hillier F. and Connors M. "Quadratic assignment problem algorithms and the location of indivisible facilities", *Management Science* **13**(1966) 42-57

70 Hubert L. "*Assignment problems inn combinatorial data analysis*", Marcel Dekker Inc., New York 1987

71 Huntley C. and Brown D. "Parallel Genetic algorithms with local search", *Computers and Operations Research* **23**(1996) $n°6$ 559-571

72 Ireland C. "An eigen-vector structure for a facility location problem and a related algorithm", Report for the *Office of Naval Research Project* $n°$ *NR*-042-267 1974

73 Kaku K. and Rachamadugu R. "Layout design for flexible manufacturing systems", *European Journal of Operational Research* **57**(1992) 224-230

74 Kaku B. and Thompson G. "An exact algorithm for the general quadratic assignment problem", *European Journal of Operational Research* **2**(1986)281-293

75 Karish S. "Nonlinear approaches for the quadratic assignment and graph partitioning problems", *Ph. D. Thesis*, Technical University Graz, Austria 1995

76 Karisch S. and Rendl F. "Lower bounds for the quadratic assignment problem via triangle decompositions", *Mathematical Programming* **71**(1995) 137-151

77 Kaufman L. and Broeckx F. " An algorithm for the quadratic assignment problem using Benders' decomposition", *European Journal of Operational Research* **2**(1978) 207-211

78 Kelly J., Laguna M. and Glover F. "A study of diversification strategies for the quadratic assignment problem", *Computers and Operations Research* **21**(1994) $n°8$ 885-893

79 Kettani O. and Oral M. "Reformulating quadratic assignment problems for efficient optimization", *IEE transactions* **25**(1993) 97-107

80 Koopmans T. and Beckmann M. "Assignment problems and the location of economic activities", *Econometrica* **25**(1957) 53-76

81 Lacksonen T. "Static and dynamic layout problem with varying areas", *Journal of the Operational Research Society* **45**(1994) $n°1$ 59-69

82 Land A. "A problem of assignment with interrelated costs", *Operations Research Quarterly* **14**(1963) 185-198

83 Laport G.. and Mercure H. "Balancing hydraulic turbine runners: A quadratic assignment problem", *European Journal of Operational Research* **35**(1988) 378-382

84 Laursen P., "Simple approaches to parallel branch and bound", *Parallel Computing* **19**(1993) 143-152

85 Laursen P., "Simulated annealing for the QAP: Optimal trade-off between simulation time and solution quality", *European Journal of Operational Research* **69**(1993) 238-243

86 Lawler E. "The quadratic assignment problem", *Management Science* **9**(1963) 586-599

87 Li Y., Pardalos P., Ramakrishnan K. and Resende M. "Lower bounds for the quadratic assignment problem", *Annals of Operations Research* **50**(1994) 387-410

88 Li W. and Smith M. "An algorithm for Quadratic assignment problems", *European Journal of Operational Research* **81**(1995) 205-216

89 Lovasz L. and Schrijver A. "Cones of matrices and set function of 0-1 optimization", *SIAM Journal on Optimization* **1**(1991) 166-190

90 Mangoubi R. and Mathaisel D. "Optimizing gate assignments at airport terminals", *Transportation Research* **19**(1985) 173-188

91 Mans B., Mautor T. and Roucairol C. "A parallel depth first search branch and bound algorithm for the quadratic assignment problem", *European Journal of Operational Research* **81**(1995) 617-628

92 Mautor T. and Roucairol C. "A new exact algorithm for the solution of quadratic assignment problem", *Discrete Applied Mathematics* **55**(1992) 281-293

93 Mirchandani P. and Obata T. "Algorithms for a class of quadratic assignment problems", presented at the joint *ORSA/TIMS* National meeting 1979 New Orleans

94 Moore J. "Optimal locations for multiple machines", *Journal of Industrial Engineering* **12**(1961) 307-313

95 Mosevich J. "Balancing hydraulic turbine runners - A discrete combinatorial optimization problem", *European Journal of Operational Research* **26**(1986) 202-204

96 Muhlenbein H. "Parallel genetic algorithms, population genetics and combinatorial optimization", in: *Proceedings of the* 3^{rd} *conference on Genetic Algorithms* (1989) 416-421

97 Muller-Merbach H. "*Optimale Reihenfolgen*", Springer Berlin-Heidelberg-New york 1970

98 Nugent C., Vollman T. and Ruml J. "An experimental comparison of techniques for the assignment of facilities to locations", *Operations Research* **16**(1968) 150-173

99 Padberg M. and Rijal M. "*Location scheduling design and integer programming*", Kluwer Academic Publishers, Boston 1996

100 Pardalos P. and Crouse J. "Parallel algorithm for the quadratic assignment problem", in *Proceedings of the Supercomputing Conference* ACM Press (1989) 351-360

101 Pardalos P., Ramakrishnan K., Resende M. and Li Y. "Implementation of a variance reduction-based lower bound in a branch and bound algorithm for the quadratic assignment problem", *SIAM Journal on Optimization* **7**(1997) $n°1$ 280-294

102 Pardalos P. and Wolkowicz H. (eds.) "*Quadratic Assignment and Related Problems*", DIMACS series in Discrete Mathematics and Theoretical Computer Science, volume **16**, 1994, American Mathematical Society

103 Parker C. "An experimental investigation of some heuristic strategies for component placement", *Operations Research Quarterly* **27**(1976) 71-81

104 Patel A., DeWald C. and Cote L. " Pre-partitioning as a means of enhancing pairwise interchange solutions to quadratic assignment problems", *Computers and Operations Research* **3**(1976) 49-56

105 Pegels C. "Plant layout and discrete optimizing", *International Journal of Production Research* **5**(1966) 81-92

106 Pierce J. and Crowston W. "Tree-search algorithms for quadratic assignment problems", *Naval Research Logistics Quarterly* **18**(1971) 1-36

107 Pomentate J. "On minimization of backboard wiring functions" *SIAM Review* **9**(1967) 564-568

108 Ramachandran B. and Pekny J. "Lower bounds for nonlinear assignment problems using many body interaction", *European Journal of Operational Research* **105**(1998) 202 -215

109 Ramakrishnan K., Resende M. and Pardalos P. "A branch and bound algorithm for the quadratic assignment problem using a lower bound based on linear programming", *Proceeding of State of the Art in Global Optimization: Computational Methods and Applications* Kluer Academic Publishers (1996) 57-73.

110 Rendl F. and Wolkowicz "Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem", *Mathematical Programming* **53**(1992) 63-78

111 Resende M., Pardalos P. and Li Y. "Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problem using GRASP", *ACM transactions on Mathematical Software* **22**(1996) $n°1$ 104-118

112 Resende M., Ramakrishnan and Drezner Z. "Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming", *Operations Research* **43**(1995) $n°5$ 781-791

113 Roucairol C. "A reduction method for the quadratic assignment problem", *Operations research Verfahren (Methods of Operations research)* **32**(1979) 183-187

114 Roucairol C. "A parallel branch and bound algorithm for the quadratic assignment problem", *Discrete Applied Mathematics* **18**(1987) 211-225

115 Sahni S. and Gonzalez T. "P-complete approximation problems" *Journal of the ACM* **23**(1976) 555-565

116 Sarker B., Wilhelm W. and Hogg G. "Backtracking and its amoebic properties in one-dimensional machine location problems", *Journal of the operational Research Society* **45**(1994), $n°9$ 1024-1039

117 Sarker B., Wilhelm W., Hogg G. and Han M. "Backtracking of jobs in one-dimensional machine location problems", *European Journal of Operational Research* **85**(1995) 593-609

118 Seehof J. and Evans W. "Automated plant layout design program", *Journal of Industrial Engineering* **18**(1967) 690-695

119 Shang J. "Multicriteria facility layout problem: An integrated approach", *European Journal of Operational Research* **66**(1993) 291-304

120 Sherali H. "The Quadratic Assignment Problem: Exact and Heuristic methods", *Ph. D. Thesis*, Georgia Institute of Technology USA 1979

121 Sherali H. and Brown E. "A quadratic partial assignment and packing model algorithm for the airline gate assignment problem", in *Quadratic Assignment and Related Problems, P. Pardalos*

and H. Wolkowicz eds. DIMACS series in Discrete Mathematics and Theoretical Computer Science, **16**(1994) 343-364 *American Mathematical Society*

122 Sherali H. and Rajgopal P. "A flexible, polynomial time, construction and improvement heuristic for the quadratic assignment problem", *Computers and Operations Research* **13**(1986) $n°5$ 587-600

123 Sherali H. and Adams W. "*A Reformulation-Linearization Technique for solving discrete and continuous nonconvex problems*", Kluwer Academic Publishers 1999

124 Skorin-Kapov J. "Tabu search applied to the quadratic assignment problem", ORSA Journal on Computing **2**(1990) 33-45

125 Skorin-Kapov J. "Extensions of a tabu search adaptation to the quadratic assignment problem", *Computers and Operations Research* **21**(1994) 855-865

126 Smith H. "A computational comparison of an improved pair assignment algorithm and a pair exchange algorithm for the quadratic assignment problem", *AD-AOP*19799, 1975

127 Steinberg L. "The backboard wiring problem: A placement algorithm" *SIAM Review* **3**(1961) 37-50

128 Taillard E. "Robust tabu search for the quadratic assignment problem", *Parallel Computing* **17**(1991) 443-455

129 Tate D. and Smith A. "A genetic approach to the quadratic assignment problem", *Computers and Operations Research* **22**(1995) 73-83

130 Timofeev B. and Litvinov V. "On the external value of a quadratic form", Kibernetica **4**(1969) 56-61 in Russian

131 Tsuchiya K., Bharitkar S. and Takefuji Y. "A neural network approach to facility layout problems", *European Journal of Operational Research* **89**(1996) 556-563

132 Vollman T. and Buffa E. "The facility layout problem in perspective", *Management Science* **12**(1966) $n°10$ 450-468

133 Vollman T., Nugent C. and Zartler R. "A computerized model for office layout", *The journal of Industrial Engineering* **19**(1968) 321-329

134 Wess B. and Gotschlich M. "Optimal DSP memory layout generation as a quadratic assignment problem", in *Proceedings of the IEEE International Symposium on Circuits and Systems* **3**(1997) 1712-1715 Hong Kong

135 West, D. H.,"Algorithm 608: Approximate solution of the quadratic assignment problem", *ACM Transactions on Mathematical Software* **9**(1983) 461-466

136 White D. "A convex form of the quadratic assignment problem", *European Journal of Operational Research* **65**(1993) 407-416

137 White D. "Strengthening Gilmore's bound for the quadratic assignment problem", *European Journal of Operational Research* **77**(1994) 126-140

138 White D. "The use of specially structured models for obtaining bounds in the quadratic assignment problem", *Journal of the Operational Research Society* **45**(1994) $n°4$ 451-462

139 White D. "Some concave-convex representations of the quadratic assignment problem", *European Journal of Operational Research* **80**(1995) 418-424

140 Wilhelm M., Ward T. "Solving Quadratic assignment problems by Simulated annealing", *IEE transactions* **19**(1987) 107-119

141 Wimmert R. "A mathematical method of equipment location", *The journal of Industrial Engineering* **9**(1958) 498-505

142 Yamada, S.,"A new formulation of the quadratic assignment problem on r-dimensional gird", *IEEE Transactions on Circuits and Systems-I Fundamental theory and applications* **39**(1992) 791-797

143 Zhao Q. "Semi-definite programming for assignment and partitioning problems", *Ph. D. Thesis*, University of Waterloo, Canada 1996

144 Zhao Q., Karish S., Rendl F. and Wolkowicz H. "Semi-definite relaxations for the quadratic assignment problem", *Journal of Combinatorial Optimization* 2(1998) $n°1$, 71-109.